VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Vývoj internetových aplikací pomocí RIA technologií

# Internet application development based on RIA technologies

2009 Bc. Jan Kovář

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2009 .....

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Michalovi Radeckému za podnětné a trpělivé vedení, a dále všem, kteří mi během práce naslouchali a přispívali drobnými nápady a podporou.

## Abstrakt

Práce se zabývá porovnáním současných technologických rámců pro vývoj bohatých Internetových aplikací. Pro toto porovnání je vybráno pět nejznámějších technologických rámců, které zastupují různé programovací jazyky a technologie. Jsou to Adobe Flex, Google Web Toolkit, JavaFX, Microsoft Silverlight a OpenLaszlo. Výsledkem je ucelený rozbor vlastností jednotlivých rámců. Srovnání je doplněno o rozbor výhod a nevýhod při použití těchto rámců spolu s doporučením pro výběr vhodného rámce při vývoji konkrétního typu RIA aplikace. Vybrané rámce jsou prezentovány jednoduchými aplikacemi, které obsahují tři základní prvky RIA aplikace: grafické komponenty, multimédia a klient-server komunikaci.

**Klíčová slova:** Web, Internet, bohaté Internetové aplikace, Adobe Flex, Microsoft Silverlight, OpenLaszlo, JavaFX, Google Web Toolkit

## Abstract

This work deals witch a comparison of current technological frameworks for development of RIA. Fiveof the most famous technological frameworks were chosen for this comparison representing various programming languages and technologies. They are Adobe Flex, Google Web Toolkit, JavaFX, Microsoft Silverlight and OpenLaszlo. The result is a comprehensive analysis of each individual framework. The comparison is completed by an anylisis of the advantages and disadvantages of using this frameworks, together with recommend for choosing an appropriate framework for developement of concrete types of RIA. The chosen frameworks are represented in simple applications which include three basic aspects of RIA: graphic components, multimedia and client-server communication.

**Keywords:** Web, Internet, Rich Internet Applications, Adobe Flex, Microsoft Silverlight, OpenLaszlo, JavaFX, Google Web Toolkit

## Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| AJAX | – | Asynchronní JavaScript a XML |
| API | – | Application Programming Interface |
| CERN | – | The European Organization for Nuclear Research |
| CLR | – | Common Language Runtime |
| CSS | – | Cascading Style Sheets |
| DB | – | datové úložiště v podobě databáze |
| DHTML | – | dynamické HTML |
| DLL | – | Dynamic-link library |
| DNS | – | Domain Name System |
| DOM | – | Document Object Model |
| Flex | – | označení pro technologii Adobe Flex |
| GTK+ | – | multiplatformní vývojová sada pro tvorbu grafických komponent |
| GUI | – | Graphical User Interface |
| GWT | – | Google Web Toolkit |
| HTML | – | Hypertext Markup Language |
| HTTP | – | Hypertext Transfer Protocol |
| HTTP | – | Hypertext Transfer Protocol Secure |
| IDE | – | Integrated development environment |
| JAR | – | Java Archive |
| JDE | – | Java Development Kit |
| JRE | – | Java Runtime Environment |
| JVM | – | Java Virtual Machine |
| LPS | – | Open Laszlo Server |
| MP3 | – | MPEG-1 Audio Layer 3 |
| MVS | – | Microsoft Visual Studio |
| MXML | – | XML pro tvorbu uživatelského rozhraní pro Flex aplikace |
| OS | – | operační systém |
| POJO | – | Plain Old Java Object |
| RIA | – | Rich Internet Application - bohatá Internetová aplikace |
| RMI | – | Remote Metod Invocation |
| RPC | – | Remote Procedure Call |
| RSS | – | Really Simple Syndication |

| | | |
|---|---|---|
| RTE | – | Runtime Environment - prostředí pro běh aplikace |
| SDK | – | software development kit |
| Silverlight | – | Microsoft Silverlight |
| SWF | – | Shockwave Flash |
| UI | – | User Interface |
| WAR | – | Web Archive |
| Web | – | označení pro World Wide Web - množinu hypertextových dokumentů dostupných přes Internet |
| Windows | – | OS Microsoft Windows |
| WPF | – | Windows Presentation Foundation |
| XAML | – | Extensible Application Markup Language |
| XML | – | eXtensible Markup Language |
| kurzíva | – | názvy proměnných, metod, tříd, značek, absolutní názvy balíčků a cest |
| uvozený text | – | přenesený význam uvozeného textu |
|  | – | logo technologie Flex |
|  | – | logo technologie GWT |
|  | – | logo technologie JavaFX |
|  | – | logo technologie Silverlight |
|  | – | logo technologie OpenLaszlo |

# Obsah

# Seznam tabulek

# Seznam obrázků

# Seznam výpisů zdrojového kódu

# 1 Úvod

Vývoj aplikací v Internetu není jednoduchá záležitost. Vývojář se musí často potýkat s nekompatibilitou své aplikace v různých prohlížečích, zejména pak rozdílným zobrazováním UI. Také výběr a spojení vhodných technologií, jak už klientských, tak na straně serveru může být v některých případech obtížné. Vývojář tak uvítá jakoukoliv pomoc, jak při samotném vývoji aplikace, tak při optimalizaci a nasazení.

Moderní RIA rámce poskytují vývojářům jednoduché řešení v podobě komplexního balíku vývojových nástrojů, technologií a postupů k snadnému vytvoření webové aplikace. Tento text byl napsán, aby tyto rámce prozkoumal, porovnal mezi sebou a poskytl tak komplexní přehled o možnostech vývoje bohatých Internetových aplikací. Hlavním účelem je poskytnout vývojáři základní představu o RIA technologiích, možnost nahlédnout pod pokličku jednotlivých rámců v podobě ilustračních příkladů a zdrojových kódů.

V první části textu je nastíněna historie a úvod do vývoje webových a bohatých Internetových aplikací. Je zde nastíněn rozdíl mezi pojmy Web 1.0 a Web 2.0 a představeny jednotlivé typy aplikací v prostředí Internetu. Hlavní cíl druhé části textu je představit pojem RIA a nejznámější zástupce RIA technologií.

Třetí část práce se zabývá samotným srovnáním vybraných technologií. Rozdělení kapitoly do tří částí poskytuje čtenáři, potažmo vývojáři, možnost získání cenných informací potřebných při rozhodování, jakou technologii pro vývoj aplikace použít. Poslední část práce se zabývá již konkrétním návrhem a implementací jednoduchých aplikací ve zvolených RIA technologiích.

# 2 Aplikace v prostředí Internetu

O významu slova internet, tedy s malým počátečním písmenem a slova Internet, se vedou dlouhodobě spory [6]. Jaký je mezi nimi rozdíl? Pojem Internet převládal od dob jeho vzniku a vyjadřoval vlastní název sítí spojených do jednoho velkého celku. Velký Internet. Pod pojmem internet tedy můžeme chápat jen technologické propojení jednotlivých domácích či firemních prvků do jedné menší sítě. Internet je potom celosvětové spojení takovýchto malých sítí.

Historie Internetu sahá do roku 1945, kdy byla publikována esej As We May Think od Vannevara Bushe [22]. V tomto textu Vannevar předpovídal budoucnost takovým technologiím, jako jsou dnes již běžné osobní počítače, Internet, Web nebo online encyklopedie. Dalšími významnými mezníky v historii Internetu jsou roky 1958 - založení organizace ARPA [17] pro výzkum nových technologií pro armádu, rok 1962 - vznik projektu počítačového výzkumu agentury ARPA, rok 1969 - první síť ARPANET sestavená ze čtyř univerzitních počítačů, 1973 - myšlenka protokolu TCP, 1983 - rozdělení ARPANETu na vojenskou a civilní část, vyvinutí DNS, 1987 - vzniká pojem nekomerční Internet, 1990 - odstřižení ARPANETu od veřejné části, 1991 - WWW v laboratořích CERN. Za dalších patnáct let obsahuje Internet již půl miliardy uzlů a více než sto milionů serverů.

## 2.1 Historie WWW

World Wide Web, WWW, neboli Web[19] je globální informační prostředí, které mohou uživatelé číst a přepisovat pomocí počítače připojenému k Internetu. Skládá se z hypertextových dokumentů, tedy takových, které jsou uloženy na nějakém serveru a které lze získat pomocí hyper odkazu, tedy odkazů uvnitř a mezi hypertextovými dokumenty. V dnešní době se již pojem Web stává pro širší veřejnost pojmem Internetové aplikace. I protokol HTTP již dnes neslouží jen k přenosu hypertextových dokumentů [24].

Web je mladší než Internet, avšak je s ním úzce spojen. V roce 1980 Sir Timothy John Berners-Lee [21] vytvořil systém Enquire [18], systém, který obsahoval myšlenky současného Webu. Neměl však být, oproti dnešnímu Webu, poskytnut veřejnosti. V roce 1989 byl CERN největším Internetovým uzlem v Evropě a Berners-Lee dostal nápad. Spojit hypertextové dokumenty, TCP protokol, DNS systém a Internet. World Wide Web byl na světě. V roce 1990 byl na světě již první webový server, NeXT Computer, první webový prohlížeč WorldWideWeb a napsány první webové stránky, které popisovaly projekt Webu a samy sebe. První veřejný webový prohlížeč byl Mosaic z roku 1993. V tomto roce byl také Web prohlášen za veřejný bez jakýchkoliv poplatků.

Podle výzkumu z roku 2001 bylo na Webu více než 550 miliard dokumentů. V roce 2002 prokázal průzkum 2,024 milionů webových stránek, že nejvíce z prozkoumaných stránek je v anglickém jazyce. Dalšími významnými jazyky jsou němčina, francouzština a japonština. V roce 2008 bylo ve známém Webu, viditelném pomocí vyhledávacích mechanismů, nejméně 68 miliard webových stránek.

## 2.2 Web 1.0

Termín Web 1.0 je označení pro dobu historie Webu, stylu a designu vytváření webových stránek před nástupem fenoménu Web 2.0. Nejlépe, a většinou to jinak nejde, se dá tento pojem popsat ve srovnání s Webem 2.0, avšak základní prvky a myšlenky by mohly být shrnuty následovně:

- Statické stránky místo dynamicky generovaného obsahu.
- Použití rámců uvnitř stránky pro oddělení obsahu.
- Téměř žádná interaktivita webové stránky a uživatele - pouze prohlížení obsahu.
- Aplikace ve Webu 1.0 jsou proprietární, tedy bez možnosti modifikace uživatelem a možnosti získání zdrojových kódů.

## 2.3 Web 2.0

Pojem Web 2.0 lze chápat jako druhou generaci vývoje a designu Webu. Koncepty Webu 2.0 vedly k vývoji a evoluci komunit, hostovaných služeb - business služeb poskytovaných druhou stranou skrze Internet a aplikací na Webu. Typickými příklady jsou stránky pro sdílení videa, wiki stránky, blogy, stránky pro setkávání lidí. Dalším znakem je folksonomie, způsob označování a sdílení obsahu Webu všemi uživateli.

Termín se poprvé objevil po konferenci O'Reilly Media Web 2.0 v roce 2004 [14]. Ačkoliv termín ukazuje na novou verzi Webu, neodkazuje se na aktualizaci žádné z technických specifikací. Ukazuje však na změny, jakými softwaroví vývojáři a koncoví uživatelé používají Web.

Webové stránky verze 2.0 umožňují uživatelům více než jen získávání informací. Uživatelé mohou vlastnit data v prostředí Webu a mít nad nimi kontrolu. Web 2.0 stránky často nabízí bohaté, uživatelsky přívětivé rozhraní založené na moderních Internetových technologiích.

Typické techniky a rysy webových stránek jsou:

- Vyhledávání - jednoduchost, s jakou se dá najít informace pomocí klíčového slova.
- Odkazy - vedou k důležitým kouskům informací; nejlepší stránky jsou nejčastěji odkazovány.
- Autorství - schopnost vytvářet a měnit obsah v průběhu času; blogy, wikipedie.
- Tagy (značky) - kategorizace obsahu pomocí, většinou jednoslovných, značek, které usnadňují vyhledávání a zabraňují vytváření předem daných kategorií.
- Signály - použití RSS; při změně obsahu webové stránky je uživatel zaregistrovaný k odběru novinek automaticky upozorněn.

| Web 1.0 byl o | Web 2.0 je o |
|---|---|
| čtení | psaní |
| jednotlivci | společenství |
| HTML | XML |
| domovských stránkách | zakládají se blogy |
| portálech | RSS |
| vlastnictví | sdílení |
| Netscape | Google |
| webových formulářích | webových aplikacích |
| vytáčeném spojení | rychlém připojení |
| ceně zařízení | ceně připojení |

Tabulka 1: Srovnání Web 1.0 a Web 2.0

Následující tabulka (tabulka 1) obsahuje výběr základních rozdílů mezi Webem 1.0 a Webem 2.0 [14].

Existuje argument, že pojem Web 2.0 nereprezentuje novou verzi World Wide Webu a pouze pokračuje ve využívání technologií a konceptů Webu 1.0. Technologie jako jsou například AJAX - asynchronní JavaScript, nenahrazují protokol HTTP, ale pouze nad něj přidávají vrstvu abstrakce. Pojem Web 2.0 nemá tedy přesně definované hranice, avšak přináší některé důležité pojmy v oblasti vývoje webových a Internetových aplikací.

## 2.4 Typy aplikací

Většina lidí si pod pojmem Internetová aplikace představí nejspíše aplikaci, stránky, které běží v prohlížeči. Internetový obchod, bankovní informační systém, emailového klienta. Něco jako applet možná někdy slyšeli, ale neví přesně co to je. A jaký je vlastně rozdíl mezi Internetovou aplikací, webovou aplikací a klasickou desktopovou aplikací?

- Desktopová aplikace - aplikace napsaná pro konkrétní operační systém a spouštěná výhradně nad tímto. Pro jednoduchost uvažujme aplikaci postavenou nad grafickým rozhraním tohoto OS.
- Internetová aplikace - využívá ke svému běhu prostředí Internetu. Na první pohled vypadají z uživatelského hlediska stejně jako desktopové. Příkladem může být emailový klient, komunikační programy, samotné WWW a další.
- Webová aplikace - aplikace poskytovaná skrze prohlížeč v prostředí Internetu.
- RIA - webové aplikace, které mají některé vlastnosti aplikací Internetových.

Každý z tohoto typu aplikace má své výhody i nevýhody. Krátce se zmiňme o Internetových a webových aplikacích.

## 2.5 Internetové aplikace

Protože Internetová aplikace neběží přímo v prohlížeči, potřebuje způsob, jakým si data z Internetu vyžádá a zobrazí. Běžně používaným a známým druhem způsobu komunikace je klient - server. Klient, který zadává požadavky a zobrazuje odpovědi, server tyto požadavky zpracovává a vrací požadovaná data. Další druh je Peer-To-Peer, zkráceně P2P. Tyto aplikace mají vlastnosti jak klienta, tak serveru. Typickým příkladem jsou telefonní hovory nebo posílání zpráv po internetu, takzvaný instant messaging. Třetím druhem může být hybridní P2P. Typickým příkladem aplikace s tímto druhem komunikace je BitTorrent. Služba přímého sdílení dat dvou a více uživatelů přes Internet, kteří však svá data nabízí prostřednictvím serveru.

Pro zobrazení dat slouží dva základní typy klientů. Tenký a tlustý. Tenký klient se stará pouze o zpracovaní a zobrazení získaných dat a zadávání požadavků na server. Je plně závislý na připojení k serveru. Tlustý klient je naopak schopen zpracovávat některá data přímo na straně uživatele. Má většinou také přístup k lokálním zdrojům dat a umí pracovat v offline módu. Také není až tak náročný na synchronizaci se serverem.

Dalšími klienty jsou hybridní klient, pro kterého se OS načítá ze sítě a chytrý klient, který kombinuje vlastnosti tenkého a tlustého klienta.

Jak bylo již řečeno výše, tento typ aplikací se uživateli může jevit stejně jako desktopová aplikace. Mezi výhody těchto aplikací patří:

- Dostupnost klienta přímo po spuštění aplikace - odpadá režie na stažení a konstrukci klienta u uživatele.
- Klasické intuitivní uživatelské rozhraní - rozložení ovládacích prvků, hlavní pracovní plocha.
- Běh aplikace offline - tato vlastnost určitě konkuruje webovým aplikacím a některým RIA aplikacím.

První bod kladných vlastností je spíše jen předpoklad. V praxi mohou být některé webové aplikace a RIA rychlejší než Internetová aplikace poskytující stejnou funkcionalitu. Práce offline na druhou stranu jasně převyšuje zbylé dva typy. Prohlížet si HTML stránky offline, psát si emailovou korespondenci "do šuplíku" a odeslat ji až bude klient online, jsou bezesporu vlastnosti a body k dobru pro Internetové aplikace.

Vlastnosti hovořící proti desktopovým aplikacím jsou následující:

- Musí se instalovat, aktualizovat - uživatel si aplikaci koupí většinou jako krabicový produkt a sám si spravuje uvedení aplikace do provozu.
- Těžko se přenáší mezi počítači, pro různé OS existují různé verze - pro některé aplikace stále nepřekonatelný problém.
- Aplikace jsou komerční, nebo pro rozšířené funkce požadují poplatek.

Díky výše uvedeným záporným vlastnostem si můžeme vytvořit srovnání s webovými aplikacemi, popsanými v následující kapitole.

## 2.6 Webové aplikace

V prvopočátcích klient-server komunikace měla každá aplikace své uživatelské rozhraní pro každý server zvlášť. Rozšíření serveru tak znamenalo většinou projít a aktualizovat také klienta na každé pracovní stanici. Oproti tomu slouží ve webových aplikacích jako univerzální rozhraní pro většinu klientů, prohlížečů, jazyk HTML a v dnešní době častěji používané XHTML. Jednotlivé stránky jsou dodávány postupně s každým požadavkem na server a uživateli se tak může zdát aplikace jako částečně interaktivní.

Webmail, neboli elektronická pošta v prohlížeči, online aukční síně, wikipedie, to vše jsou webové aplikace. A jak to vše začalo?

### 2.6.1 Historie

V roce 1995 společnost Netscape Communications Corporation přišla s novým skriptovacím jazykem na straně klienta nazvaným JavaScript, který umožnil vývojářům přidat do statických HTML určitou dávku interaktivity. Rok poté, v roce 1996, společnost Macromedia, která je dnes součástí společnosti Adobe Systems, představila Flash. Přehrávač instalovaný do prohlížeče jako plugin, pomocí kterého lze do webové aplikace vložit animované video. To umožnilo použít skriptovací jazyk pro interakci s uživatelem na straně klienta. Rozšířené a známé jsou především Flashové hry a reklamy.

V roce 1999 byl koncept webové aplikace představen pod rouškou jazyka Java a jeho Servlet specifikací. V té době se stále čekalo na objevení technologie AJAX. To přišlo až v roce 2005 a webové aplikace, jako například elektronická pošta Gmail od společnosti Google, Inc., tak začaly svou dlouhou cestu Internetem.

### 2.6.2 Struktura

Rozhraní webových aplikací nabízí téměř stejnou funkcionalitu jako aplikace desktopové či Internetové. Díky jazyku Java, JavaScriptu, DHTML, Flash a dalších technologií lze na klientovi například malovat, přehrávat audio, kontrolovat pohyb myši a stisk kláves. Další technologie jako je drag and drop, táhni a pusť, jsou dnes již také podporovány.

Aplikace jsou obvykle rozděleny do logických kusů nazývaných vrstvy. Každá vrstva má svou určitou roli. Nejčastější vrstvy u webových aplikací jsou:

- Databázová vrstva - běží na serveru a obsahuje funkcionalitu obsluhy databází.
- Aplikační vrstva - běží také na serveru a stará se o zpracování požadovaných dat od klienta, které získává z databázové vrstvy a vrací je klientovi.
- Prezentační vrstva - zajišťuje zpracování a zobrazení dat u klienta.

Tyto tři vrstvy mohou být podle potřeb rozloženy do více specifických menších vrstev. Například mezi aplikační a databázovou vrstvou může být integrační vrstva, která zapouzdří volání aplikační a prezenční vrstvy do databázové vrstvy. Aplikace tak volá knihovní funkce z integrační vrstvy a ta se postará o korektní načtení dat z databáze. Výhodou je, že klient nemusí znát implementaci databázové vrstvy a pouze volá jednotný kód pro přístup dat. Při změně databáze se tak nemusí přepisovat celá střední vrstva a vymění se jen knihovny pod integrační vrstvou.

### 2.6.3 Vlastnosti

Výhodou webových aplikací je, jak již bylo zmíněno, že by měly běžet a chovat se stejně navzdory použitému OS. Oproti psaní aplikace pro každý OS zvlášť je tak aplikace napsána jen jednou a distribuována téměř kdekoliv. Avšak různorodý způsob vykreslování HTML, CSS a DOM jádry prohlížečů, může způsobit někdy drobné, někdy však markantní rozdíly v prezentaci webové aplikace. Standardizace prohlížečů, potažmo způsob vykreslování uvedených prvků, je však téměř nemožná.

Mezi výhody můžeme uvést:

- Přenositelnost mezi OS - jedna aplikace běží, až na některé výše popsané nesrovnalosti, téměř ve všech prohlížečích.
- Automatická aktualizace - nová verze přichází k uživateli spolu s novými funkcemi bez toho, aniž by se to uživatel musel dozvědět.
- Nižší nároky na klienta - aplikace většinou potřebují ke svému běhu menší diskový prostor.

Oproti tomu mají webové aplikace i své slabé stránky:

- Problém jednotného chování a zobrazení - popsáno výše.
- Stále závislost na spojení se serverem - aplikace nejsou schopny běžet v offline režimu.
- Vyšší nároky na bezpečnost - díky použití skriptovacích jazyků je potřeba dbát zvýšené bezpečnosti při psaní klientského kódu.

Při vývoji webových aplikací se dnes již hojně používá podpůrných frameworků, které obsahují širokou řadu možností podpory vývoje. Od poskytnutí implementace vícevrstvé architektury, podpory validací, předchystaných knihoven skriptů, až po implementaci podpory offline režimu.

Jako řešení implementace standardního chování pro všechny prohlížeče se nabízí otázka použití Java appletů nebo Flashe. V dnešní době, kdy již většina prohlížečů tyto technologie v podobě pluginů podporuje, nemusí být nasazení aplikace na klienta příliš složité. Hlavní výhoda těchto technologií je však v tom, že má programátor větší

kontrolu nad vytvořením jednotného uživatelského rozhraní díky jasné specifikaci zpracování jazyka Java nebo Flashe a tak může překlenout specifické problémy konfigurace prohlížečů.

Při použití těchto technologií se však již mírně mění architektura aplikace. Díky podobnosti s tradičními klient-server aplikacemi, s jakýmsi tenkým klientem v podobě appletu nebo Flashe, se vedou spory o to, zdali nezařadit tyto modifikované webové aplikace mezi RIA. Jaký je mezi nimi rozdíl si uvedeme v následující kapitole.

# 3 RIA - bohaté Internetové aplikace

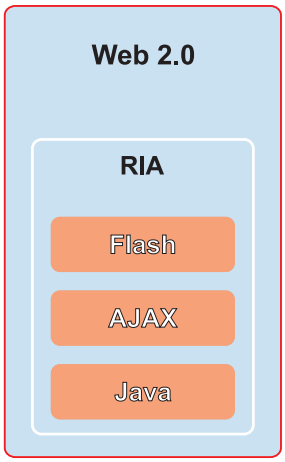
RIA jsou webové aplikace, které mají rysy a funkcionalitu tradičních Internetových aplikací. Jejich standardní prostředí je webový prohlížeč obohacený o plugin ve kterém aplikace běží. Mezi těmito aplikacemi jsou však i výjimky, které místo pluginů a svých runtime prostředí využívají čistě prostředí prohlížeče a jsou postaveny zcela nebo z větší části na JavaScriptu.

Pojem RIA byl představen v roce 2002 společností Macromedia [2]. Dokument v úvodu obsahuje ohlédnutí za vývojem webových aplikací od roku 1990 a dále se zabývá vývojem a využitím škálovatelných bohatých Internetových aplikací a vše ilustruje na svém produktu Macromedia Flash MX. Pojem RIA však může být stejně matoucí jako pojem Web 2.0. S postupem času se vedle Flashe představil také AJAX, giganti jako Microsoft nebo Sun Microsystems nechtěli zůstat pozadu a technologie jako Microsoft Silverlight nebo JavaFX byly na světě.

V dnešní době jsou RIA známé pro své bohaté uživatelské prostředí, líbivou grafiku. Uživatelům se přiblížily natolik, že jsou již téměř k nerozeznání podobné desktopovým aplikacím. Známá aplikace YouTube, některé internetové obchody, emailoví klienti, prezentace známých osobností, vědecké a statistické aplikace, hry a spousta dalších jsou důkazem oblíbenosti RIA nejen u uživatelů.

Aplikační rámce jednotlivých technologií pro vývoj RIA poskytují softwarovým vývojářům rozšíření jejich znalostí o vývoji webových aplikací a nabízí jejich rozšíření právě o dynamické či multimediální prvky. Vývoj takovéto aplikace tak často sdružuje nejen samotné programátory, ale také grafiky, databázové a další specialisty.

Dá se říci, že svět bohatých Internetových aplikací je v současnosti asi tak široký pojem, jako množství technologií ve Webu 2.0, i když jsou RIA právě jednou z jeho součástí (obrázek 1).




Obrázek 1: RIA technologie jako součást Webu 2.0

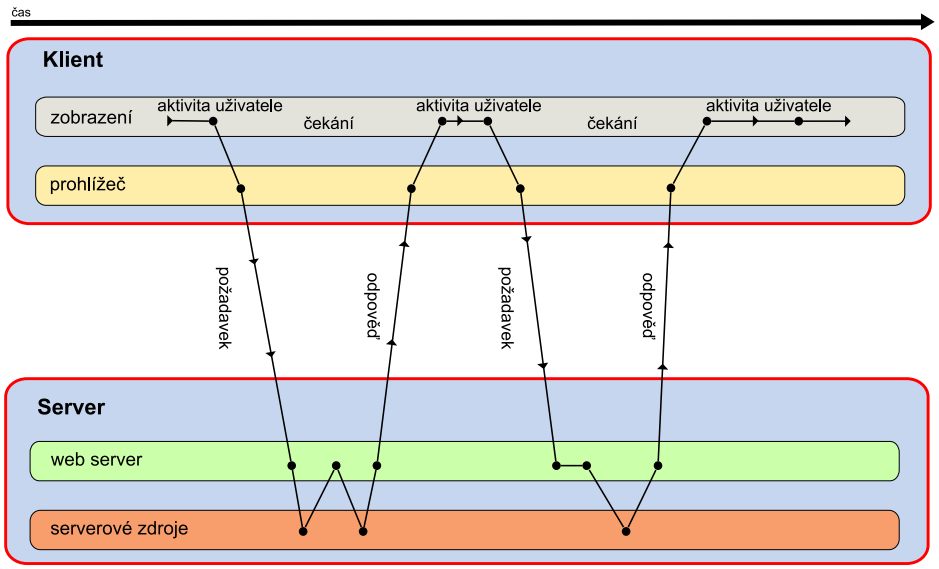
V následujících odrážkách je uvedeno upřesnění pojmů RIA aplikace a RIA technologie, které budou v textu dále používány.

- RIA technologie - souhrn technologických postupů a prostředků pro vytvoření RIA aplikace. Jedná se především o název technologie, základní programovací jazyk, či koncept komunikace klient-server a základní strukturu rozložení zdrojových kódů.
- RIA aplikace nebo jen RIA - aplikace postavená nad konkrétní RIA technologií.

## 3.1 Vlastnosti RIA aplikací

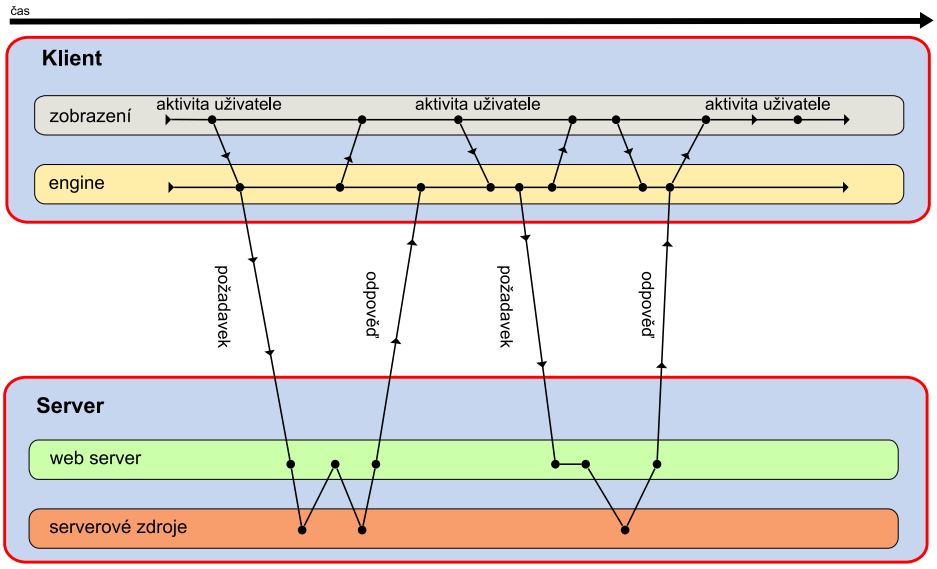
Oproti webovým a Internetovým aplikacím mají RIA jeden zásadní rozdíl. Ačkoliv oba přístupy používají ke komunikaci se serverem princip request-response, tedy požadavek a odpověď, RIA tento princip využívá odlišně. Klientská aplikace nekomunikuje se serverem přímo, ale využívá prostředí, ve kterém běží, jako prostředníka. Až ten teprve přeposílá požadavky klienta na server a zpracovává odpovědi. Navíc se vše děje asynchronně.

Webová aplikace je na začátku stáhnuta na klientský počítač a zobrazena. Uživatel poté s aplikací pracuje. Při potřebě aktualizace dat je nejdříve zavolán prohlížeč, který předá požadavek na data serveru. Po dobu zpracování požadavku serverem, například při načítání dat z databáze či jiného časově náročného procesu, musí uživatel čekat. Až po vrácení dat serverem a aktualizací v prohlížeči může uživatel pokračovat v práci. Schéma komunikace klasické webové aplikace je zobrazeno na obrázku 2.




Obrázek 2: Synchronní komunikace webových aplikací

Místo přímého volání serveru RIA požádá o data mezivrstvu, klientský engine, který vytvoří požadavek a pošle jej serveru. Těchto požadavků může zpracovat více podle




Obrázek 3: Asynchronní komunikace RIA

potřeb uživatele a zobrazení. Celá aplikace se tedy nemusí znovu načítat po každém požadavku. Engine si pamatuje jednotlivé požadavky a podle jednotlivých odpovědí serveru aktualizuje příslušné komponenty. Celá komunikace tak zůstává uživateli skrytá a po dobu jednoho volání serveru může pracovat s jinou částí aplikace. Tento způsob komunikace je výhodný například u zpracování většího množství dat, například z databází nebo multimédií. Schéma komunikace RIA aplikace, která využívá engine jako prostředníka, je zobrazena na obrázku 3.

Z výše uvedeného je tedy patrné, že RIA se od webových aplikací liší především v:

- Asynchronní komunikaci mezi serverem a klientem,
- umožňuje funkcionalitu na straně klienta díky klientskému enginu,
- s aplikací lze pracovat v čase efektivněji.

Na závěr této kapitoly je uvedena rekapitulace jednotlivých typů aplikací a jejich srovnání (tabulka 2). Tato tabulka ukazuje na některé základní rozdíly mezi jednotlivými typy technologií.

- Zatímco u desktopové a Internetové aplikace je potřebná interakce uživatele při instalaci, webové aplikace a RIA jsou instalovány automaticky při jejich spuštění. U webové aplikace je proces instalace většinou omezen na stažení aplikace do prohlížeče. Některé RIA však potřebují kromě stažení na klienta instalaci podpůrného engine, většinou instalovaného jako plugin do prohlížeče.

| vlastnost | typ aplikace | | | |
|---|---|---|---|---|
| | desktop | Internetová | webová | RIA |
| instalace klienta | vždy | vždy | bez instalace | částečně |
| nezávislost na OS | ne | ne | ano | ano |
| závislost na síťovém připojení | ne | částečně | ano | ano |
| práce offline | výhradně | většinou ano | ne | částečně |
| | | | | |
| nároky na klienta | vysoké | vysoké | nižší | nižší |
| automatické aktualizace | ne | ano | ano | ano |
| | | | | |
| intuitivní ovládání | ano | ano | ne | částečně |
| modifikovatelnost | ne | ne | ano | ano |

Tabulka 2: Typy aplikací a jejich vlastnosti

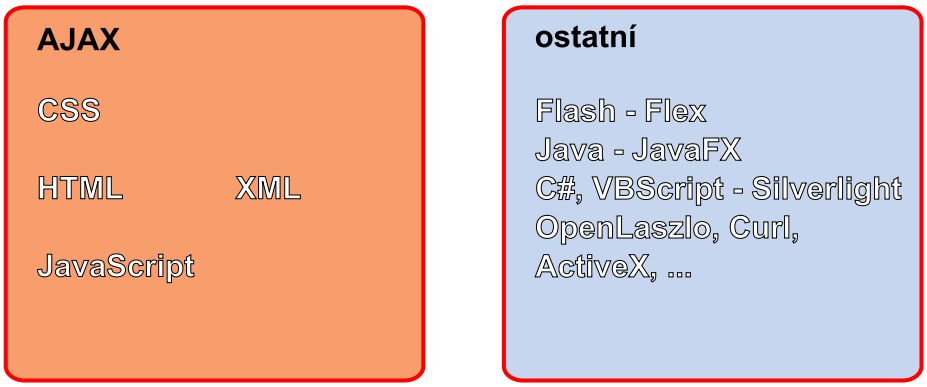
- Webové a RIA aplikace lze spustit na libovolném počítači s příslušným pluginem. I přes absenci některých verzí pluginů je jejich přenositelnost větší než instalovaných, na OS závislých aplikacích.
- Desktopové aplikace nevyžadují připojení do sítě, Internetové si s dočasným výpadkem připojení poradí v offline režimu, avšak webové aplikace a RIA bez připojení k síti nemohou běžet.
- S výše uvedeným bodem souvisí možnost práce offline. Některé RIA jsou schopny zajistit offline režim za cenu snížení funkcionality.
- Jelikož webové aplikace a RIA propagují na klienta jen zobrazení s minimální funkcionalitou, jsou jejich nároky na výpočetní výkon klienta podstatně nižší než u desktopových a Internetových aplikací.
- Nové verze jsou dostupné automaticky pro typy aplikací se síťovým připojením. Desktopové aplikace jsou odkázány na ruční aktualizaci. Rozšíření ve smyslu přidání funkcionality je také snazší u webových aplikací a RIA.
- Uživatelské rozhraní klasických aplikací využívá standardního rozložení tlačítek a ovládacích prvků. Webové aplikace a RIA již díky své podstatě nemají definováno jednotné uspořádání ovládacích a zobrazovacích prvků. RIA se však snaží o přenesení možností bohatých Internetových aplikací do prostředí prohlížeče a tak definuje vlastní sadu ovládacích prvků, které se dají spojit do celku připomínajícího prostředí desktopu.

  Desktopové aplikace jsou vhodné pro maximální využití výpočetního výkonu spojeného s prací nad lokálními daty. Uživatel může předpokládat stejné chování aplikace na různých počítačích, avšak ztrácí možnost její aktualizace. Internetové aplikace

představují rozšíření desktopových aplikací o možnost práce se vzdálenými daty. Webové aplikace poskytují maximální nezávislost na klientském prostředí za cenu snížené uživatelské přívětivosti a funkcionality. RIA spojují výhody již zmíněných technologií a přináší do světa webových aplikací nový rozměr - funkcionalitu desktopových aplikací a uživatelskou přívětivost.

## 3.2 Přehled současných technologií

Současné RIA technologie mohou být rozděleny do dvou směrů (obrázek 4). Technologie na jedné straně jsou zcela nebo částečně postaveny nad technologií AJAX. Zatímco zbylé technologie využívají prostředky a technologie jako jsou Flash, Java a vlastní programovací či skriptovací jazyk.



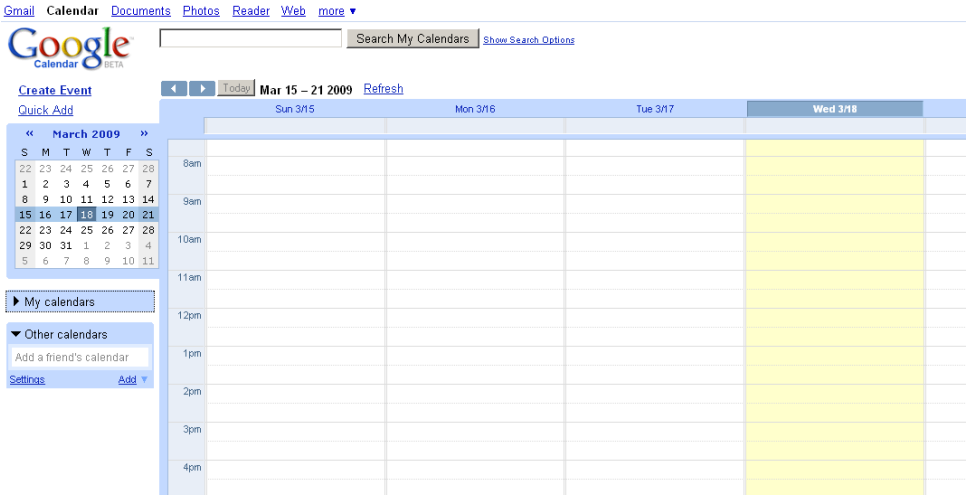

Obrázek 4: Rozdělení RIA technologií

Tvorba AJAXové RIA je založena na propojení JavaScriptu, XML, HTML a CSS. Programátor tak z větší části využije znalosti z programování klasických webových aplikací. Aplikace postavené nad HTML a CSS pak uživateli poskytnou kompaktní konzervativní uživatelské rozhraní webových aplikací.

Snad nejznámějším propagátorem a uživatelem AJAXu je Google. Mezi jeho nejznámější aplikace patří Google Maps, Gmail, aplikace pro sdílení fotografií Picasa, Google Calendar a další aplikace z Google rodiny.

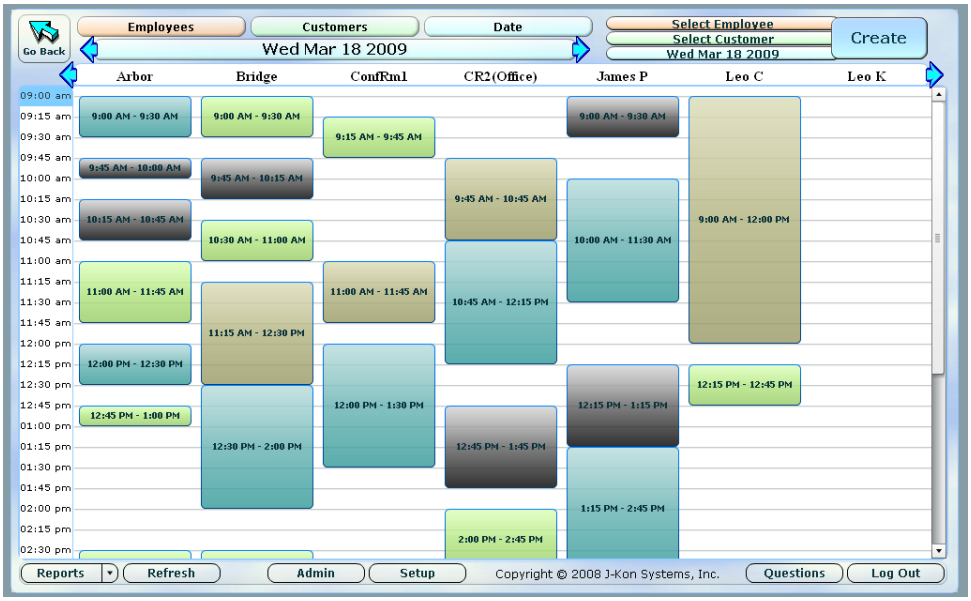
Ukázka na obrázku 5 je část aplikace Google Calendar. Aplikace je tvořena především pomocí JavaScriptu a CSS stylů.

Pro aplikace, které jsou postavené nad druhou skupinou technologií, tedy Flashi, aplikace technologií OpenLaszlo nebo Silverlight, poskytují uživateli vyšší vizuální požitek. Všechny tyto tři zmíněné technologie obsahují prostředky pro tvorbu bohatých vizuálních prezentací.

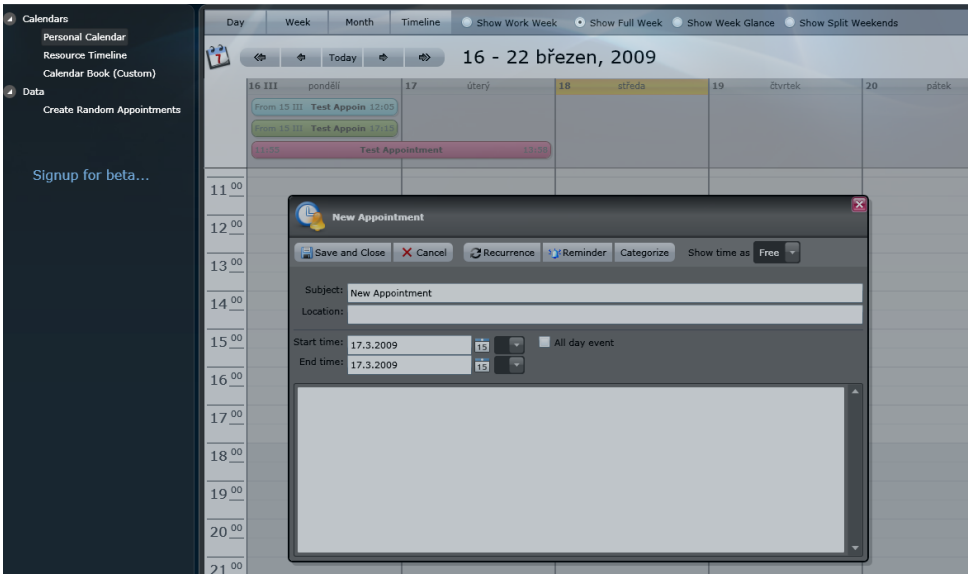



Obrázek 5: Google kalendář

Pro srovnání jsou přiloženy prezentace postavené nad Flash a Silverlight technologií (obrázek 6 a obrázek 7). Uživatelské rozhraní těchto aplikací je obohaceno o barevné přechody, stínování a další barevné prvky.




Obrázek 6: Kalendář schůzek napsaný ve Flashi




Obrázek 7: Plánovač v technologii Silverlight

Další technologie, které se dají zařadit mezi RIA technologie patří například: Backbase a Curl, které jsou založeny na značkovacím jazyku obdobnému HTML a JavaScriptu, dále Mozilla Prism a Google Gears, které rozšiřují webové prohlížeče a poskytují offline podporu webovým aplikacím, stejně tak jako Adobe AIR. Dále například Rich Ajax Platform, což je spojení Eclipse IDE, AJAXu, technologie JFace - sady nástrojů pro vytváření uživatelského rozhraní a sady grafických komponent. Výčet může pokračovat názvy jako jsou REBOL, Cappuccino, WPF pro tvorbu bohatých uživatelských rozhraní, nebo na XML založený jazyk XUL - XML User Interface Language.

Mezi technologie, které nejsou přímo RIA, avšak jsou součástí některé z nich, můžeme také zařadit pojmy Java Web Start a applety.

Java Web Start je nástroj, který umožňuje spuštění Java aplikací přímo z webového prohlížeče. Uživatel musí znát pouze odkaz na aplikaci, kterou si následně stáhne a Java Web Start se postará o spuštění, automatickou aktualizaci a další. Tato technologie také díky tomu, že je aplikace stažena kompletně na klientovi, umožňuje spuštění aplikace v offline módu. Toho například využívá RIA technologie JavaFX, která umožňuje spuštění svých aplikací právě pomocí Java Web Startu.

JavaFX aplikace lze také spustit pomocí appletů. Applet je komponenta, která běží v kontextu jiného programu a skrze kterou je spuštěna klientská aplikace. Nejznámější je Java applet, který podporuje spouštění aplikací napsaných právě v jazyce Java. Díky některým svým vlastnostem, jako je nápadná podobnost s pluginy Flashe a Silverlightu, zabezpečení, poskytování a omezování některých funkcí na straně klienta a z větší části právě díky JavaFX technologii, jsou Java applety stále využívány.

## 3.3 Vybrané technologie

Následující kapitola obsahuje stručné představení pěti nejznámějších technologií, které stojí v současné době na pomyslném vrcholu mezi RIA technologiemi a jsou základem pro většinu moderních bohatých Internetových aplikací. Jsou také kombinací různých technologií, programovacích jazyků a metod vývoje. Pomocí jejich srovnání tak bude možné vytvořit si ucelenější představu o současných možnostech vývoje RIA aplikace. Pořadí hodnocení a představení aplikací nemá vliv na jejich popularitu, je voleno pro přehlednost podle abecedy.

### 3.3.1 Adobe Flex 3

Adobe Flex představuje souhrn technologií pro vývoj a nasazení bohatých Internetových aplikací založených na platformě Adobe Flash. Vývoj Flex aplikací je postaven na XML orientovaném značkovacím jazyce MXML pro tvorbu uživatelského rozhraní a ActionScriptu, který přináší interakci mezi komponentami a tvoří jádro aplikace. ActionScript je také standardem pro Flash přehrávač a není tedy divu, že právě on se stará o práci na pozadí líbivého uživatelského rozhraní, které běží právě ve Flash přehrávači.

Od roku 2004, kdy byla vydána první verze Flexu s číslem 1.0 a později 1.5 a první licencí, která byla prodána, se Flex hodně změnil. Jednak přibývaly nové verze podpůrných technologií, například nová verze ActionScriptu v roce 2005, avšak hlavní změna se udála právě ve strategii obchodní. Zatímco verze 1.0 byla kompletně zpoplatněna, dnes je verze 3.2 poskytována zdarma pod jednou z veřejných licencí. Zajímavým bodem je také od verze 2.0 vývoj a představení IDE postaveného nad Eclipse. Bohužel, toto je od svého prvopočátku pro širokou veřejnost stále zpoplatněno. V roce 2009 chystá společnost Adobe novou verzi s pořadovým číslem 4.

### 3.3.2 Google Web Toolkit 1.5

GWT je implementační rámec, který poskytuje podporu vývojářům při tvorbě rozsáhlých, na AJAXu založených aplikací. Uživatelské rozhraní aplikace napsané jako asynchronní Java je převedeno GWT překladačem do optimalizovaného JavaScriptu, který je poté natažen do webového prohlížeče. GWT překladač optimalizuje výsledný JavaScript a aplikace je tak rychlejší a méně náchylná na chybu než při ručním psaní skriptů. Aplikace napsané v GWT nepotřebují pro svůj běh žádný plugin, je nutné pouze povolit JavaScript v prohlížeči.

Za svou relativně krátkou dobu se GWT stal zajímavým konkurentem velikánů od Adobe nebo Microsoftu a to už minimálně tím, že se pevně drží JavaScriptu oproti ostatním, na míru ušitým jazykům. Během dvou let, od počáteční verze 1.0, bylo vydáno pět verzí GWT. Za tuto dobu se stihl GWT zbavit počátečních chyb a vykrystalizovat do ucelené podoby v aktuální verzi 1.5. Nejvíce změn a nových vlastností bylo provedeno mezi verzemi 1.3 až 1.4. Verze 1.5 byla přelomová. GWT od té doby podporuje rysy Javy 1.5 jako jsou generické datové typy, výčtové typy, anotace a další. Aktuální verze je 1.5.3 a v plánu na rok 2009 je pak verze 1.6.

### 3.3.3 JavaFX 1.1

JavaFX platforma je založená na standardním JRE a deklarativním jazyce pro vývoj uživatelského rozhraní známého pod názvem JavaFX Script. Jelikož je JRE rozšířené na mnoha platformách, je JavaFX považována až na výjimky OS Linux a Solaris za multiplatformu. JavaFX je podporována i pro mobilní zařízení s Java Micro Edition.

Historie této platformy je krátká. Sun Microsystems představil JavaFX veřejnosti na jaře v roce 2007. O rok později, na jaře 2008, společnost oznámila uvedení na trh na podzim téhož roku a do mobilních zařízení na jaře roku 2009. Celý rok 2008 tak mohli vývojáři testovat ukázkovou verzi. V prosinci roku 2008 byla zveřejněna verze 1.0 a od února roku 2009 je dostupná verze 1.1, kde součástí instalačního balíčku je jak SDK, vývojové prostředí NetBeans, pluginy pro podporu vývoje uživatelského rozhraní do programů Adobe Photoshop a Illustrator a již zmíněná podpora pro mobilní zařízení pod názvem JavaFX Mobile.

### 3.3.4 Microsoft Silverlight 2.0

Silverlight je plugin pro většinu webových prohlížečů, postavený nad částí .NET platformy a WPF. Pro vytváření uživatelského rozhraní Silverlight aplikací se používá deklarativní, nad XML postavený jazyk XAML. Pro vývoj jádra aplikace lze využít širokou škálu jazyků, které jsou podporovány .NET platformou. Hlavními jsou C#, Visual Basic.NET, Delphi, J#.

Do roku 2007 byl Silverlight znám pod zkratkou WPF/E, rozepsáno Windows Presentation Foundation/Everywhere. Verze 1.0, která byla vydána v témže roce byla postavena čistě nad XAML a JavaScriptem a nepotřebovala ke svému běhu žádný plugin, obsahovala pouze základní sadu komponent pro tvorbu uživatelského rozhraní, základní podporu pro přehrávání zvuku a malou podporu pro zpracování událostí. Místo verze 1.1 byla v roce 2008 uvolněna verze 2.0, která již obsahuje komplexní knihovnu komponent pro uživatelské rozhraní, podporu vláken, podporu databází a zejména podporu pro širokou řadu programovacích jazyků podporovaných .NET platformou. Tato verze současně obsahuje a je zpětně kompatibilní s verzí 1.0

### 3.3.5 OpenLaszlo 4.2

OpenLaszlo je open source framework pro vývoj RIA postavený nad skriptovacím programovacím jazykem LZX a OpenLaszlo serverem (LPS). LZX je podobně jako MXML a XAML deskriptivní jazyk postavený nad XML a JavaScriptem. LPS je Java servlet, který kompiluje LZX aplikace do binární podoby pro různé runtime prostředí. Konkrétně se jedná o Flash a DHTML.

Historie technologie OpenLaszlo sahá do roku 2000, kdy se pod názvem Laszlo Prezentation Server objevují první zmínky o této technologii. Mezi lety 2001 až 2005 proběhl vývoj prvních verzí frameworku a testovacích aplikací a v roce 2005 byly uvolněny první verze, již přejmenovaného frameworku, a to verze 3.0 a 3.1. V lednu roku 2009 již existuje verze 4.2.0.1.

# 4 Srovnání vybraných technologií

V předchozí kapitole bylo spolu s úvodem do problematiky RIA uvedeno pět nejrozšířenějších RIA technologií. V následujících odstavcích budou tyto technologie porovnány a to z pohledů:

- Obecných vlastností - základní architektura, obsah vývojové sady a její dostupnost, systémové požadavky a další.
- Funkcionality - to znamená jaké prostředky pro řešení základních implementačních problémů technologie nabízí; tvorba UI, získávání dat ze serveru, multimédia.
- Uživatelského - jakým způsobem si vytvořená aplikace, potažmo právě UI a kvalita zpracování, nebo její rychlost získají uživatele.

Srovnání technologií je provedeno formou bodového hodnocení. Rozdělení vah jednotlivých skupin vlastností je rozděleno přibližně na třetiny s přidáním většího důrazu na funkcionalitu. Zvýšení bodového hodnocení právě funkcím bylo zvoleno s ohledem srovnání technologií pro vývojáře, který se bude rozhodovat pro konkrétní řešení jeho problému a nebude ho až tak zajímat, jak se k němu přes okrajové vlastnosti dostane.

Hodnocení je ve větší míře subjektivní. Jelikož například u některých základních vlastností, jako jsou jazyk nebo architektura, neexistuje jednoznačná referenční hodnota, je těmto vlastnostem automaticky přiřazeno nejvyšší bodové hodnocení. Naopak u vlastností, kde existuje přibližná referenční hodnota, je tato uvedena a vlastnost je hodnocena podle toho, do jaké míry se této hodnotě přiblíží. Stupnice byla vybrána od jedné do deseti, pro lepší možnost přirazení rozdílů testovaných vlastností. Za každou skupinu vlastností je přiřazeno technologii souhrnné bodové hodnocení, podle kterého je na první pohled vidět její rozdíl mezi ostatními. Rozdíl mezi výslednými známkami, tedy souhrnným bodovým hodnocením všech kategorií, jednotlivých technologií je pak zřetelným výsledkem celého srovnání.

Některá dílčí klíčová bodová hodnocení jsou vždy uvedena na konci jednotlivých kapitol. Srovnávací tabulka je uvedena v příloze.

## 4.1 Obecné vlastnosti technologií

Kategorie obecných vlastností obsahuje především takové, které nesouvisí s čistým programováním a při vývoji slouží pouze jako reference nebo je při jejich aplikaci potřeba znát pouze základní práci s OS. Takovým příkladem je vývojové prostředí a jeho instalace či obsah vývojové sady.

### 4.1.1 Základní programovací jazyk

Tato vlastnost se řadí mezi ty, které nemají referenční hodnotu, jelikož každý programátor preferuje jiný styl programování a má v oblibě jinou skupinu programovacích či skriptovacích jazyků. Nicméně je důležité si na tomto místě jednotlivé jazyky představit.

Pro Flex jsou základní dva stavební prvky jazyk MXML a ActionScript. MXML je deklarativní značkovací jazyk pro tvorbu UI. Jako většina jazyků odvozených od XML potřebuje pro svůj běh nějaké runtime prostředí pro vykreslení UI prvků a podpůrný skriptovací jazyk pro provázání a interakci UI prvků se zbytkem aplikace. Tímto je u Flexu ActionScript, který je zároveň základním stavebním kamenem Flash aplikací. Flex aplikace jsou distribuovány pomocí SWF. MXML je pouze syntaktickou nadstavbou ActionScriptu. Při kompilaci Flex aplikace do SWF je nejdříve kompletní MXML kód převeden do ActionScriptu a ten následně do SWF bytekódu.


Google web toolkit je založen čistě na jazyce Java, který překladač přeloží do JavaScriptu. Jednotlivé UI prvky jsou definovány přímo ve zdrojovém Java souboru klienta a syntaxe je podobná jako při vytváření desktopové Java aplikace v knihovně Swing. Další technologií je neméně důležité HTML a DOM struktura webové stránky, pomocí níž lze přesně definovat, kde budou jednotlivé grafické prvky na stránce umístěny. Ruku v ruce s popsanými technologiemi jde CSS pro ještě preciznější stylování jednotlivých grafických komponent.

Deklarativní JavaFX Script je základ technologie JavaFX. Jedná se o kompilovaný, statický typovaný, deklarativní, skriptovací jazyk postavený nad Java platformou. Tedy jazyk, jehož zdrojové kódy jsou kompilovány do bytekódu pro JVM a typová kontrola je prováděna při kompilaci zdrojových kódů. Díky tomu, že je JavaFX Script postaven nad Java platformou, může být do JavaFX Script svázán s jiným Java nebo JavaScript kódem a prvky uživatelského rozhraní rozšířit o standardní komponenty z knihovny Swing. JavaFX Script je při kompilován do bytekódu pro JVM, tudíž může běžet v libovolném prostředí s JRE.


Další z rodiny značkovacích jazyků pro tvorbu UI je jazyk XAML. Spolu s jazyky rodiny CLR, nejčastěji C# nebo Visual Basic.NET, tvoří základ aplikací Silverlightu. Zdrojový XAML kód je kompilován do bytekódu (.baml), který je v případě Silverlightu zabalen do DLL knihovny a distribuován spolu s dalším obsahem na klienta v XAP archivu. Pro doplnění, XAP archiv je pouze jakási přepravka pro přenos dat ze serveru do Silverlight runtime. Lze jej otevřít libovolným ZIP archivátorem. Aplikace přeložená do bytekódu potřebuje pro svůj běh runtime, který v případě Silverlightu je plugin v prohlížeči.


OpenLaszlo aplikace jsou psány nad XML postaveným jazykem LZX, který je kompilován pro různé runtime prostředí. Aplikace, které jsou určeny pro běh přímo v prohlížeči jsou kompilovány do DHTML. Pro Flash přehrávač lze LZX kód kompilovat do SWF a to verzí SWF8 a SWF9. Pro interakci UI prvků s uživatelem a další logiky na straně klienta je použit JavaScript.

Výše uvedený seznam představuje výčet základních jazyků pro vývoj klienta pro konkrétní technologii. Jsou postaveny nad XML a JavaScriptem, či jejich modifikovanou verzí. Proto je u většiny z nich možnost rozšíření právě o tyto dva jazyky a s nimi souvisejícími technologiemi. Jelikož klienti získávají data ze serveru různým způsobem, technologií a jazyků pro vývoj serverové části je mnoho, avšak každá RIA technologie se snaží využít serverovou technologii z rodiny svého vydavatele.

Ukázka zdrojových kódů je spolu s detailnějším popisem jednotlivých implementací v kapitole 5.2.

### 4.1.2 Architektura

Pod tímto pojmem bude u jednotlivých technologií představen souhrn vlastností, které se týkají struktury klientské části aplikace a její natažení do prohlížeče.


Flex je framework, aplikační rámec, jako součást Flash platformy pro vývoj RIA aplikací, který se v základu skládá z MXML a ActionScriptu. Je vázán na runtime prostředí Flash přehrávače. Pro překlad do Flash podoby jsou potřeba podpůrné knihovny a kompilátor.

MXML a ActionScript jsou spolu s použitými knihovnami nejprve přeloženy do SWF souboru. Při distribuci přes webový prohlížeč je odkaz na aplikaci uveden jako součást načítacích funkcí v JavaScriptu v úvodní HTML stránce. Při načtení stránky je na serveru nalezen příslušný SWF soubor, natažen do Flash přehrávače a spuštěn.

GWT je složeno v minimální formě ze dvou částí. Část JRE knihoven, emulovaných pro GWT a to konkrétně *java.lang* a *java.util* a sadě GWT UI knihoven. Pro doplnění souvislostí lze počítat do GWT platformy také překladač Javy do JavaScriptu. Spuštění aplikace je pak zřejmé. Zdrojové Java soubory jsou překladačem přeloženy do JavaScriptu, který je spolu s webovými stránkami distribuován na klienta. Jelikož je celá aplikace kompilovaná do JavaScriptu, je aplikace spuštěna po jeho zpracování prohlížečem.


JavaFX platforma obsahuje ve svém základu specifické JavaFX runtime prostředí, nejčastěji applet, dále obecné prvky, které tvoří základ aplikace pro všechny cílové platformy a dále specifické prvky pro desktopové, mobilní a jiná zařízení. Aplikační rámec již jen zastřešuje a poskytuje možnost spojit uvedené prvky dohromady.

Vytvoření a nasazení aplikace se skládá ze dvou částí. Zdrojové soubory JavaFX Scriptu jsou přeloženy do bytekódu a zabaleny do JAR archivů. Pokud je aplikace spouštěna přes prohlížeč, běží pak jako applet, který je natažen díky pomocnému runtime engine ve formě JavaScriptu. Ten je stažen spolu s prvním spuštěním JavaFX aplikace. Kompilátor také automaticky vytvoří deskriptor pro Java Web Start.

Jádro pro tvorbu aplikací Silverlightu tvoří podmnožina .NET platformy a prezentační framework. Část .NET platformy obsahuje základní komponenty pro integraci dat, síťové služby, základní třídní knihovny a CLR knihovny. Prezentační vrstva obsahuje komponenty a služby zajišťující zobrazení a zpracování událostí od UI komponent. Součástí této vrstvy je i XAML.

Díky tomu, že Silverlight aplikace potřebuje ke svému běhu Silverlight runtime, je aplikace distribuována podobně jako Flex aplikace. Odkaz na XAP archiv je uložen na klientovi a při načtení webové stránky je archiv stažen a zpracován. Drobným vylepšením je stahování na požádaní. To znamená, že lze aplikaci rozdělit do více částí a například větší datové soubory jako obrázky či video sekvence lze načítat zvlášť.

Architektura aplikace pro OpenLaszlo se skládá z OpenLaszlo runtime knihoven (ORL), které jsou kompilovány do všech aplikací a poskytují runtime služby. Dále prezentační vrstvy, která se stará především o vykreslování UI komponent. Tím, že aplikace běží přímo buď přímo v prohlížeči jako DHTML nebo Flash prezentace, řadí se někdy k architektuře Flash přehrávač a webový prohlížeč.

Způsob distribuce aplikace je naprosto rozdílný od předchozích. Na aplikaci se klient dotazuje přímo a to konkrétně na zdrojový LZX soubor. Server požadovaný soubor najde, spolu s obrázky a dalšími runtime knihovnami aplikaci zkompiluje a vrátí klientovi. Buď jako prezentaci pro Flash přehrávač nebo jako data pro zobrazení pomocí DHTML. Aplikaci lze také distribuovat již předkompilovanou bez využití mezivrstvy LPS, tím však aplikace ztratí některé výhody, které LPS přináší.

### 4.1.3 SDK

Další přehled je obsah SDK. Není překvapením, že většina technologií obsahuje základní sadu knihoven, kompilátor a dokumentaci.


Flex SDK obsahuje: základní sadu knihoven, nástroje pro vývoj aplikace z příkazové řádky, příklady, kompletní dokumentaci a runtime prostředí pro běh vytvořených aplikací.

GWT je na tom podobně. Stažené SDK obsahuje: knihovny a kompilátor. Dokumentace je omezena na základní popis knihoven generovaný z JavaDocu. Zajímavostí je však možnost nechat si vygenerovat strukturu projektu pro Eclipse IDE. GWT v základu také nabízí runtime prostředí, v podobě vlastního jednoduchého prohlížeče s podporou logování.


Základní sada knihoven a kompilátor zdrojových kódů nechybí ani v distribuci JavaFX SDK. To také obsahuje offline dokumentaci a tutoriál pro vytvoření základních aplikací a několik předpřipravených demo aplikací.

 Silverlight je na SDK relativně chudé. Obsahuje pouze sadu knihoven a dokumentaci, kterou však lze spustit jen s pomocí vývojového nástroje.

 Oproti Silverlightu je OpenLaszlo distribuce o hodně bohatší. Obsahuje všechny základní prvky SDK což jsou knihovny, příklady, dokumentace. Specifické na distribuci je, že obsahuje i serverovou část a s ní i překladač pro Flash aplikace. Je to pochopitelné, server překládá zdrojové soubory LZX a posílá na klienta. Tedy SDK obsahuje i tuto část.

Další vlastností SDK u všech technologií je, že jsou zdarma, což je pozitivní informace. U vývojových nástrojů už to tak nemusí být, jak bude uvedeno dále. Další otázka jsou veřejné zdrojové kódy. Adobe poskytuje veřejné zdrojové kódy pro Flex SDK a to pouze pro framework, kompilátor a debugger. GWT nechá nahlédnout pod pokličku knihoven, JavaFX zatím neposkytuje veřejné kódy kromě možnosti zjistit kompletní gramatiku a syntaxi. Silverlight poskytuje zdrojové kódy pro některé UI prvky. Jediné OpenLaszlo je zcela veřejné.

### 4.1.4 Systémové požadavky

Každá aplikace má určité požadavky na systém. Většinou je to kompatibilita mezi OS. Dále pak pomocnými knihovnami, které ke svému běhu potřebuje. U RIA jsou tyto požadavky rozděleny na dvě části. Jedna zahrnuje OS a druhá webový prohlížeč spolu s potřebou instalovat pomocné knihovny nebo runtime prostředí v podobě pluginu. Třetím nepovinným požadavkem, jelikož jsou RIA zaměřeny spíše na klienta, jsou nároky serverové strany. Co vše RIA potřebují ke svému běhu je uvedeno v následujícím seznamu.

 Podle dokumentace lze Flex vyvíjet na operačních systémech Windows XP, Windows 2000, Mac OS verze 10.4.x, OS Linux Red Hat, SUSE a OS Solaris. Aplikace napsané ve Flexu lze spustit ve všech prohlížečích, které podporují Flash přehrávač. Ten je také jedinou podmínkou pro spuštění aplikace. Flash přehrávač je dostupný pro přední webové prohlížeče, což jsou Internet Explorer, Mozilla Firefox, Netscape a Opera.

Vývoj na serveru, respektive nasazení aplikace na serverové části nemá žádná omezení a řídí se pouze specifikacemi daného webového serveru.

 GWT potřebuje ke svému běhu na klientovi JRE minimálně ve verzi 1.4.2 a vyšší. Podporované OS jsou Windows XP, Windows 2000, Mac OS a OS Linux s podporou GTK+ knihovny minimálně ve verzi 2.2.1. Jako hostitele může použít GWT aplikace prohlížeče Internet Explorer, Mozillu Firefox, Safari nebo Operu. Jedinou podmínkou pro běh GWT aplikace je v prohlížeči povolený JavaScript.

Obdobně jako u Flexu není při vývoji potřeba server. Vše je překládáno přímo do JavaScriptu a pro testování je v SDK připraven testovací prohlížeč.

JavaFX klade na vývojové prostředí nemalé požadavky. Potřebuje JDK verzi 6 update 7 a s ním spojené JRE. Podporuje pouze Windows XP a Windows Vista v 32-bitové verzi a Mac Os. JavaFX zatím nelze vyvíjet na OS Linux. Podporovanými prohlížeči jsou Internet Explorer, Mozilla Firefox a Safari. Jako jediný požadavek na rozšířené prostředí je JRE pro spuštění appletů a Java Web Start technologii a engine v podobě skriptu, který se stáhne na klienta spolu s prvním spuštěním JavaFX aplikace.

Jak již bylo řečeno v sekci 4.1.2, aplikace je složena z JAR archivů, které jsou staženy a spuštěny v JRE. Serverová strana proto musí podporovat spouštění Java aplikací.

Silverlight podle očekávání podporuje nejvíce verzí Windows. Jsou to XP, 2000, Vista a dokonce poslední beta verzi Windows 7. Dále je možno Silverlight vyvíjet na Mac Os a existuje i podpora pro OS Linux skrze Moonlight Project [13]. Prohlížeči pro Silverlight mohou být Internet Explorer, Mozilla Firefox nebo Safari. Do všech však musí být doinstalován Silverlight plugin.

Silverlight aplikace nelze vyvíjet z příkazové řádky a dost často se k vývoji používá komplexní IDE. To v sobě již obsahuje server, který je se Silverlightem asociován. Obecně lze aplikaci vyvíjet nad libovolným webovým serverem s povolenou Silverlight asociací.

Pro vývoj OpenLaszlo aplikace lze jako OS použít Windows XP, Linux s verzí jádra 2.6 a vyšší nebo Max OS. Technologie se směle hlásí k prohlížečům Internet Explorer 7, Microsoft Firefox 3 a Safari 3 s tím, že by OpenLaszlo aplikace měly fungovat i pod nižšími verzemi. Pokud je aplikace kompilována jako Flash prezentace, musí prohlížeč obsahovat Flash přehrávač plugin a to ve verzi minimálně 8 nejlépe však 9.

Specifické podmínky musí splňovat serverová část, pokud jsou v aplikaci použity média, komunikace pomocí SOAP nebo XML-RPC. Tyto služby poskytuje právě LPS, který potřebuje JRE verze 1.5 a vyšší a Java Servlet kontejner podporující specifikaci Java Servletu nejméně ve verzi 2.2

Aplikace všech výše uvedených technologií lze distribuovat jako webovou aplikaci. To znamená vytvořit příslušný WAR soubor a nainstalovat pod libovolný webovým server. Klasickým příkladem může být Apache Tomcat server, k dnešnímu dni verze 6. Ukázka takovéto distribuce a jednotlivých archivů je v kapitole 5.2.

### 4.1.5 Vývojové prostředí

V kapitolách 4.1.1 až 4.1.4 bylo uvedeno, z čeho se aplikace jednotlivých technologií skládají a co je potřeba k jejich vývoji. V této sekci je popsáno, pomocí jakých nástrojů lze takovou aplikaci vytvořit a kde je lze sehnat.

Flex platforma zdarma, není tomu tak již s podporou vývojových nástrojů. Ačkoliv lze Flex aplikaci vytvářet pomocí zdarma dostupných textových editorů, kompilovat zdrojové kódy přes příkazovou řádku, vytvářet distribuční archivy pomocí volně dostupných buildovacích nástrojů, většina vývojářů dá dnes přednost komplexnímu IDE. Adobe nabízí pro vývoj Flex Builder IDE, postavené nad platformou Eclipse. To má určitě nespornou výhodu, jelikož Eclipse je jeden z nejrozšířenějších vývojových nástrojů. Co však nepotěší programátora "samotáře", je jeho cena. Adobe poskytuje na IDE třicetidenní bezplatnou licenci, poté je však uživatel nucen si produkt zakoupit. Malou útěchou může být školní licence, která je zdarma.

IDE je dostupné přímo na webových stránkách Adobe jak pro Windows, tak pro Mac OS. Dále se IDE podle obsahu dělí na Standard a Professional. Stažení a nainstalování lze provést podle typu připojení k Internetu do třiceti minut.

IDE obsahuje standardní nabídku funkcí jako je inteligentní doplňování kódu, nástroje pro ladění, profiler, automatické testování a velice zajímavě řešené a rozsáhlé prostředí pro vývoj UI. IDE také spolupracuje s Adobe Creative Suite 3 pro vytváření vlastních grafických prezentací a komponent.

Google poskytuje společně s GWT podporu vývoje aplikací přímo v Eclipse. Vývojář má možnost pomocí jednoduchého příkazu vygenerovat kompletní strukturu a projekt pro Eclipse IDE. Vygenerovaný projekt lze do Eclipse jednoduše importovat a plně tak využít funkce, které Eclipse nabízí. Eclipse IDE je zdarma a dostupné přímo na webu jak GWT, tak autora IDE. Čas potřebný ke stažení a instalaci je přibližně stejný jako u Flexu Builder IDE, tedy třicet minut.

Dalšími nástroji pro vývoj GWT aplikací jsou IntelliJ IDEA a editory grafické části GWT aplikace GWT designér a VistaMax IDE. Všechny tři zmíněné nástroje jsou k dispozici po zaplacení určitého poplatku.

JavaFX aplikaci lze vyvíjet kompletně z příkazové řádky. Přímo na webových stránkách JavaFX je však spolu s SDK zdarma ke stažení NetBeans IDE, které obsahuje přímou podporu pro vývoj JavaFX aplikací. NetBeans IDE obsahuje kompletní podporu právě pro JavaFX aplikaci, at' už ve formě UI builderu, debuggeru, profileru nebo také ve formě automatických aktualizací jednotlivých prvků JavaFX platformy.

Stažení a instalace trvá přibližně hodinu a kromě NetBeans IDE existuje i plugin pro Eclipse.

Jelikož Silverlight SDK neobsahuje nástroje pro vývoj aplikace, je nejlepší řešení stáhnout zdarma přímo od autora MVS. Jelikož IDE zahrnuje kompletní podporu pro vývoj .NET aplikací, poskytuje tak možnost vytvořit Silverlight prezentaci a přímo k ní přidat Webové služby či propojit s jiným prvkem z rodiny .NET. Pro

vývoj Silverlight aplikace je dále potřeba doinstalovat do IDE některé pluginy pro vývoj web aplikací a samozřejmě Silverlight SDK.

Existuje i IDE postavené nad Eclipse platformou, které obsahuje kompletní sadu nástrojů pro vývoj Silverlight aplikace. Komunikuje i s MVS a některé části aplikace, jako je například implementace Webových služeb je řešena právě v MVS.

Pro vývoj líbivého UI a rozšíření standardních knihovních UI komponent existuje nástroj Expression Blend [11], který je však zpoplatněn.

OpenLaszlo aplikace, stejně jako GWT, lze vyvíjet v libovolném dostupném textovém editoru. Pro zjednodušení však existuje plugin do Eclipse IDE, který poskytuje podporu vývoje jak UI komponent a jejich rozvržení v aplikaci, tak aplikační logiky.

Plugin lze získat z webu OpenLaszlo a stažení a instalace netrvá déle než hodinu. IDE je možno stáhnout buď jako plugin do stávající instalace Eclipse IDE, nebo jako kompletní balíček. Pro vývoj a testování aplikace je potřeba mít nainstalován LPS.

Aplikace vytvořené ve výše uvedených IDE lze převést do podoby webové aplikace. Některé technologie spolu s IDE podporují přímo export do WAR archivů, u některých je potřeba balit soubory ručně nebo pomocí skriptu. Další zajímavostí a výhodou nejspíše pro Java vývojáře je, že pro všechny technologie existuje alespoň minimální možnost vyvíjet aplikace v Eclipse IDE. To se tak spolu s MVS a NetBeans IDE stává hlavním vývojovým nástrojem srovnávaných technologií.

### 4.1.6 Dokumentace

Nebýt dokumentace, vzorových příkladů a komunity "zapálených" vývojářů, mohl by být vývojář začínající s určitou technologií zmaten, v horším případě od technologie odrazen. I lidé ze zmíněné komunity a různých diskusních fór nemusí být vždy konkrétní a spíše podle známého slovního spojení, které odkazuje programátora na přečtení příslušných řádek manuálu, je vývojář odkázán právě na dokumentaci a vzorové příklady. Srovnání, jak bohatá a přehledná dokumentace u jednotlivých technologií je, následuje.


Adobe poskytuje pro Flex dokumentaci k API a kompletní popis jazyka a jeho syntaxe v rozsáhlé příručce programátora. K tomu lze na webových stránkách Adobe najít sbírku vystavených demo a ukázkových aplikací. Po zaregistrování lze různé otázky ohledně Flex technologie probrat na rozsáhlém diskusním fóru. Pro rychlý začátek vývoje lze také využít předpřipravených tutoriálů, které rozebírají základní postupy tvorby jednotlivých částí aplikace. Nechybí ani seznam a představení UI komponent.

Podobně jako Flex je pro GWT dostupná podrobná API dokumentace ve formátu JavaDoc. Dále dokumentace, která je dostupná online přímo na webových stránkách

GWT, obsahuje kompletní tutoriál vývoje GWT aplikace, dále kompletní příručku programátora, popis vývojových nástrojů, FAQ (Frequently Asked Questions - často kladené dotazy) sekci a nechybí ani přehled UI komponent a diskusní fórum.

JavaFX kromě vygenerované API dokumentace, menší FAQ sekce a několika předdefinovaných tutoriálů nabízí už jen celkem rozsáhlé diskusní fórum v rámci diskusí mateřského Sun fóra. Oblast dokumentace doplňuje sekce s předem připravenými demo aplikacemi.

Podrobná Silverlight dokumentace je součástí rozsáhle MSDN (Microsoft Developer Network) dokumentace. Dokumentace je přehledně členěná do sekcí a obsahuje jak základní seznámení s technologií Silverlight, tak řešení jednotlivých technologických postupů a to včetně zdrojových kódů a odkazem na náhled zkompilovaného výstupu. Na webových stránkách projektu Silverlight jsou k dispozici tutoriály popisující konkrétní problematiku ve formě video prezentací. Nechybí ani rozsáhlé, do více kategorií dělené, diskusní fórum.

Dokumentace OpenLaszlo technologie je řešena formou jednoduchých statických webových stránek. Pouze referenční příručka obsahuje kompletní seznam prvků platformy spolu se zdrojovými kódy a vygenerovaným výstupem. Příručka programátora je však přehledná a rozsáhlá a řeší základní možnosti vývoje OpenLaszlo aplikace. Nechybí ani instalační příručka a příručka pro nasazení aplikace. Seznam UI komponent je jednoduchý a stručný. Diskusní fórum je rozsáhlé, umístěné v rámci vlastních webových stránek. Seznam demo aplikací obsahuje odkaz na vlastní prezentaci základních stavebních kamenů s možností náhledu výstupu po kompilaci, dále pak na několik aplikací, které využívají OpenLaszlo technologii.

Rozšiřujícím kritériem pro srovnání aplikací z pohledu dostupnosti informací je, jak snadné je podle dokumentace či odkazů v Internetu získat informace o implementaci video přehrávače. Kromě technologie GWT, která, jak bude uvedeno dále, nepodporuje v základu přehrávání videa, byla dostupnost informací uspokojivá. Flex, Silverlight a OpenLaszlo mají informace k přehrávání videa přímo ve své dokumentaci a JavaFX poskytuje návod jak přehrávat video sekvence v jednom ze svých tutoriálů.

### 4.1.7 První kroky

Dokumentace všech technologií jsou více či méně podobné a skládají se většinou ze základních prvků jako je API dokumentace, základní tutoriál a programátorská příručka, diskusní fórum a minimální množství ukázkových aplikací. V následující a poslední kategorii obecných vlastností je uveden časový odhad seznámení s technologií a vývoje základu aplikace. Tento čas odráží přehlednost a dostupnost jednotlivých informací právě podle dokumentace a ukázkových příkladů a přibližné znalosti junior programátora bez větších znalostí s vývojem webových či RIA aplikací.

Jednotlivé sledované oblasti jsou:

- Seznámení s technologií - od prostudování systémových požadavků, stažení a instalace IDE až po připravení k vývoji. Zahrnuto je i seznámení se syntaxí jazyka.
- Hello World aplikace - snad každý programovací jazyk či technologie poskytuje jednoduchý návod jak vytvořit úvodní aplikaci, která vypíše či zobrazí text Hello World. Tímto se zabývala druhá část.
- Nastudování CRUD - v dnešní době rozšířený typ aplikací: create, read, update, delete. Aplikace, která načte data z databáze či jiného datového zdroje a zobrazí výsledek uživateli, typicky nějaký seznam dat v tabulce. Testování proběhlo na načítání dat ze serveru, tedy R části.
- Implementace video přehrávače - v tomto případě se jedná o nastudování a implementace přehrávání videa. U technologií, které nemají ke komponentě pro přehrávání videa přidaná ovládací tlačítka, jejich implementace. Výsledný čas neobsahuje přípravu a převod videa do jednotlivých požadovaných formátů.

Porovnání výše uvedených oblastí jednotlivých technologií je uvedeno v následujícím seznamu.

Časová náročnost seznámení s Flex technologií je přibližně 4 hodiny. Za tu dobu programátor získá znalosti o vývoji Hello World aplikace, jejíž vývoj mu bude trvat asi 20 minut. Za 2 hodiny je schopen s již získanými zkušenostmi nastudovat a implementovat jednoduché načtení dat. U této technologie je nejrychlejší způsob, jak se naučit získávat data, použít jako datový zdroj XML soubor. Konečně přehrávání videa lze napsat za přibližně 2 hodiny.


GWT svou jednoduchostí dává prostor naučit se jej velmi rychle. Oproti Flexu lze GWT prostudovat v čase kolem 2 hodin. Hello World aplikace vznikne do 10 minut a přibližně 1 hodinu stráví programátor nad implementaci CRUD aplikace. Jelikož do GWT nelze jednoduše implementovat přehrávání videa, je tento bod sloučen s předchozí implementací s důrazem na implementaci komunikace klient - server.

V rozmezí 3 až 4 hodin se pohybuje čas potřebný na prostudování Silverlight technologie. Jako u ostatních technologií se čas pro vytvoření první aplikace pohybuje na hranici 20 minut. Implementace načítání dat zabere přibližně 2 hodiny. Prostudování a implementace přehrávání videa je možné stihnout do 3 hodin.


Pro prozkoumání Silverlight technologie bude junior programátor potřebovat, stejně jako u Flexu, 4 hodiny. Za 20 minut napíše první Hello World a další 2 hodiny bude studovat a implementovat načítání dat. Tentokrát lze využít přímo podporu pro databáze a spojit aplikaci například s Microsoft SQL Serverem. Čas potřebný k implementaci video prezentace by neměl překročit 3 hodiny.

Relativně rychle se naučí programátor technologii OpenLaszlo. Prostudování mu zabere zhruba 3 hodiny avšak do 2 hodin stihne začínající vývojář napsat CRUD aplikaci a za stejný čas napíše i přehrávání videa. Jednoduchá Hello World aplikace mu nezabere více než 20 minut.

Z výše uvedeného srovnání lze soudit, že každou RIA technologii lze prostudovat během jednoho víkendu. Širší náhled si dokáže zkušenější programátor udělat během jednoho či dvou týdnů.

Na závěr kapitoly o obecných vlastnostech jsou uvedeny tři nejúspěšnější RIA technologie podle dosaženého bodového hodnocení.


Podle výše uvedeného stupně vítězů má nejlepší výsledek GWT. Je to dáno především nejnižším časem potřebným k nastudování základů a implementace technologie. Na druhém místě, s relativně malou ztrátou se umístila technologie OpenLaszlo, která má společně s na třetím místě umístěnou technologií Flex také relativně malé časové nároky na prostudování.

## 4.2 Funkční vlastnosti

V první srovnávací části byly uvedeny vlastnosti, které budou zajímat potencionálního zájemce o technologii nejvíce. V následující části budou srovnány tyto doplňující vlastnosti:

- Komponenty - co je obsahem knihoven. UI komponenty, základní stavební a utilitní třídy.
- Multimédia - podpora zpracování audia, videa, 2D a 3D grafiky.
- Zdroje dat - jak lze získat data ze serveru.
- Validace a bezpečnost - jakým způsobem probíhá validace zadávaných dat a jak je zabezpečen jejich přenos mezi klientem a serverem.
- Sandbox - obecná omezení prostředí, ve kterém RIA aplikace běží.

### 4.2.1 Vlastní komponenty

Nejspíše nejvíce pozornosti, společně s prostudováním obecných vlastností, bude věnováno vizuální podobě aplikace, vytvořené v dané technologii. UI komponenty jsou jejím základem.


Knihovny pro vývoj Flex aplikace obsahují více než sto UI komponent, včetně grafů, podpory animací, stylů a dalších. Dá se říci, že Flex je velmi dobře vybaven pro vývoj bohaté vizuální Internetové aplikace. V základních knihovnách programátor také najde podporu pro komunikaci se serverem, zpracování událostí, utilitní třídy pro správu barev a fontů a mnoho dalších.

O něco méně UI komponent obsahuje ve svém základu GWT, také díky omezení převodu Java UI prvků do JavaScriptu a jen částečné emulaci standardních Java knihoven. V základu však nechybí sada UI prvků jako tlačítka, formuláře či tabulka. Dále knihovny obsahují základní sadu tříd a rozhraní pro asynchronní komunikaci se serverem. Dále pak například třídy pro podporu lokalizace a testování.


Může se zdát, že nejméně předchystaných prvků najde programátor v knihovnách JavaFX platformy. Avšak kromě základních balíčků, které obsahují třídy pro asynchronní a http komunikaci, zpracování dat a samozřejmě základní syntaxi jazyka, najde programátor v JavaFX bohatou knihovnu pro vytváření grafiky. Ta je velmi rozsáhlá a obsahuje širokou podporu pro zpracování 2D grafických prezentací, včetně kreslících prvků, stínování a různých geometrických tvarů. Kromě toho lze do JavaFX přidat emulací libovolnou komponentu z knihovny Swing, která je potomkem třídy *javax.swing.JComponent*.

Základní podmnožina .NET technologie obsahuje kompletní podporu pro vývoj RIA aplikace. Od standardních UI prvků, přes základní knihovny s podporou kolekcí, jazyka, zpracování dat a multimédií až po podporu pro klient-server komunikaci.

OpenLaszlo nezůstává pozadu s bohatou nabídkou knihoven pro zpracování klient-server komunikace, dat ze strany serveru, jejich zobrazení na klientovi. Rovněž podporuje animace a zpracování audia a videa. Zajímavým rozšířením je knihovna pro zpracování JavaScriptu a HTML. V grafických knihovnách je to beta verze knihovny grafů.

Všechny technologie podporují v základu širokou paletu UI prvků. Představení některých z nich je uvedeno v příloze B.

### 4.2.2 Multimédia

Bohatá Internetová aplikace by jistě nebyla tak bohatá, kdyby nedokázala přehrát video snímek či oblíbenou muziku. Co a v jaké kvalitě přehrají jednotlivé frameworky je obsahem následujícího srovnání.

Podporovaným audio formátem Flexu je známý formát MP3. Video je však potřeba převést do formátu FLV (Flash Video), což může být pro vývoj multimediální aplikace z počátků problém.

Ve Flexu lze jen s obtíží napsat grafický editor podobný malování. Existuje však komponenta s názvem Canvas, na kterou lze přidávat jiné UI prvky pomocí souřadnic $x$ a $y$. Pro zobrazení 3D projekcí je potřeba využít externího nástroje. Z několika je vybrán například open source projekt Papervision3D[15].

GWT jako ve svém základu nepodporuje zdaleka tak mnoho multimediálních funkcí jako ostatní technologie. Přeci jen do JavaScriptu se multimédia převádí hůře než do předem připravených multimediálních frameworků. Pro projekt GWT však existují podpůrné nástroje v podobě wrapperů. Konkrétně pro audio se jedná o knihovnu *gwt-voices*. Pro video je třeba sáhnout hlouběji do znalostí programování webových aplikací a využít možnosti propojit výslednou HTML stránku s externím přehrávačem. GWT nepodporuje nativně přehrávání videa.

Pro 2D grafiku lze využít komponentu Canvas, podobně jako v Java knihovně AWT. Tvorbu 3D animací GWT zatím neumí.

Základním formátem pro JavaFX video je FLV. Engine pro jeho zpracování je v JavaFX nezávislý na OS. Pro aplikace určené pro Windows je možné použít audio a video formáty podporované přehrávačem Windows Media Player. Pro bezproblémové použití audio záznamu jak na OS Windows tak Mac OS je použit opět formát MP3.

2D grafika je podporována standardními knihovnami, bohužel pro 3D grafiku existuje pouze omezené API na prostorové transformace.

Silverlight, jako produkt společnosti Microsoft, podporuje jak formát MP3, tak formát WMA (Windows Media Audio). Video sekvence jsou podporovány ve formátech WMV (Windows Media Video). Silverlight také pro přenos podporuje jak protokol HTTP tak streamování pomocí protokolu MMS (Multimédia Messaging Service) nebo RTSP (Real Time Streaming Protocol).

Podpora 2D grafiky je u Silverlightu omezena na sadu knihovních funkcí, které dokáží upravit komponenty nebo vytvořit nové. Jak již bylo zmíněno, pro podporu 2D prezentací existuje grafický nástroj Expression Blend. 3D grafiku Silverlight v základu nepodporuje, ovšem pomocí nástroje Kit3D [10] lze do Silverlight aplikací přidat něco vizuálních efektů.

OpenLaszlo podporuje všudepřítomný audio formát MP3. Video je podporováno ve formě FLV formátu, někdy také zabaleného do formátu SWF. Je to celkem pochopitelné, jelikož jedno z runtime prostředí OpenLaszlo aplikací je právě Flash přehrávač, který formáty FLV a SWF podporuje. Protokoly pro přenos audio a video dat jsou HTTP a RTMP (Real Time Messaging Protocol) pro streamování.

Pomocí knihovních funkcí lze docílit modelování základních 2D animací a lze vytvořit aplikaci, ve které se dá kreslit. 3D animace OpenLaszlo v základu nepodporuje, lze jich však docílit pomocí 2D transformací.

Z výše uvedeného seznamu vyplývá, že je při vývoji multimediální aplikace potřeba externího nástroje pro převod audia a videa mezi jednotlivými formáty. Naštěstí existuje několik takových volně dostupných aplikací. Za zmínku stojí nástroj pro příkazovou řádku: ffmpeg [3].

### 4.2.3 Zdroje dat

Většina moderních aplikací pracuje s daty dynamicky. Načte je od uživatele, uloží do externího úložiště a na požádání je načte a vrátí uživateli zpět. Typů datových úložišť může být více. Od klasických databází, přes souborové systémy až po dokumenty XML. Většina RIA aplikací přistupuje k datovým zdrojům na serverové straně z důvodu omezení sandboxem. S načítáním dat tedy také souvisí forma komunikace mezi klientem a serverem.

Jak si zajišťují data jednotlivé technologie, popisuje následující seznam.

Přímý přístup do DB Flex neumožňuje. Pokud Flex aplikace potřebuje načíst data, využije k tomu nejlépe Webovou službu, data z XML načtená pomocí HTTP protokolu nebo použije RPC. Dále Flex technologie umožňuje volat například JavaBeany, EJB (Enterprise JavaBeans), nebo ColdFusion komponenty použitím LiveCycle Data Services. Z předchozího plyne, že Flex aplikace lze propojit se serverovou stranou napsanou v PHP, ASP.NETu nebo Javě.

GWT je omezená na klientovi pouze JavaScriptem. K DB se lze připojit pouze ze serverové strany. Jelikož je serverová strana GWT aplikací psána většinou v Javě, stačí na serverové straně implementovat připojení ke zvolené DB pomocí dostupného ovladače a data posílat na klienta pomocí RPC. Což je také základní prostředek pro klient-server komunikaci. Mezi další metody lze zařadit možnost použít JSON (JavaScript Object Notation) nebo XML přes HTTP. GWT rovněž umožňuje volat veřejné Webové služby a to právě pomocí JSON.

JavaFX podobně jako ostatní technologie používá k získávání dat ze serveru Webové služby. A to pomocí JSON, nebo JAX-WS (Java API for XML Web Services), tedy API pro propojení na Javy a Webových služeb, které jsou založené nad

XML. Jelikož XML Webové služby využívá i Silverlight, je možné propojit JavaFX s Webovou službou napsanou v .NETu. Místo RPC používá JavaFX volání, nikoliv procedur ale metod, takzvané RMI, což je obdoba RPC avšak pro Javu.

Základní formou komunikace klient-server pro Silverlight jsou Webové služby. Jejich tvorba je v základu podporována v Microsoft Visual Studiu. Kromě toho Silverlight podporuje RPC a pomocí technologie LINQ (Language INtegrated Query) lze v Silverlight aplikaci zpracovat data z XML, Microsoft SQL Serveru nebo ADO.NET datasety.

OpenLaszlo podporuje načítání dat z DB přes nějakého prostředníka, který dokáže generovat XML data. Tím může být na straně serveru Java Servlet, JavaServer Page (JSP), CGI (Common Gateway Interface) skript, PHP skript a další. Druhým způsobem, jak zpracovávat data, je definovat je přímo na klientovi. To však pro dynamickou změnu není příliš vhodné. OpenLaszlo klienta lze propojit se serverem také pomocí Webových služeb. Častá a relativně jednoduchá implementace je také pomocí RPC volání.

### 4.2.4 Zpracování formulářů - validace

Data zadaná uživatelem mohou obsahovat plno chyb, které by při zpracování bránily korektnímu výpočtu či běhu aplikace. Rovněž je vhodné o takto vzniklé, byť neúmyslně, chybě uživatele informovat. K tomu slouží u většiny technologií validace, tedy nástroj či způsob, pomocí kterého jsou přijatá data zpracována a ověřena jejich validnost, tedy korektnost, či oprávněnost pro další zpracování.

Validace by měly být psány podle určitých principů tak, aby dostatečně chránily jak uživatele, tak aplikaci od chybně zadaných dat a na druhou stranu, aby příliš nenarušovaly uživatelovu práci. V základu jsou to tyto principy:

- 1. Upozornění uživatele na chybu jakmile nastane, ne až po potvrzení vkládaných dat.
- 2. Upozornění na chybu okamžitě - uživatel je informován o validnosti dat ještě před ukončením jejich zadávání.
- 3. Nechat uživatele pracovat - uživatel může editovat data i při výskytu chyby, nemá možnost však data odeslat.
- 4. Presumpce neviny - zobrazit varování pouze pokud uživatel může tuto chybu opravit. Rovněž nezobrazovat chybu při výchozím nastavení komponent.


Flex obsahuje pro výše uvedené zásady kompletní řešení validací na klientské straně a to v podobě vlastního validátoru. V základní sadě tříd nalezne programátor validátory pro kontrolu řetězců, datumů, čísel, regulárních výrazů. Dále například

validátory pro kontrolu formátu emailu, poštovního čísla, telefonního čísla či čísla kreditních karet. Validátory se dají použít v kombinaci s ActionScriptem a lze tak vytvořit širokou škálu vlastních validací.


Pro validaci GWT aplikací, konkrétně klientské části existuje podpora ve formě anotací. Validační anotace pro GWT jsou však psány jak pro klientskou, tak pro serverovou část. Výhodou je jednotný systém validací mezi klientem a serverem, nevýhodou však je problém znovupoužitelnosti zdrojových kódů díky zanesení anotacemi. Existují projekty, které se toto snaží vyřešit, nicméně validace v GWT aplikacích nejsou tak přímočaré, jako u předešlého Flexu.

Další možnost je validovat data až na serveru. Avšak při tomto způsobu jsou porušena některé základní principy validací. Minimálně přibude režie na komunikaci klient-server.


Programátor JavaFX aplikací bude odkázán na vlastní pečlivost. Technologie ani knihovny neobsahují mechanismus, jak data od uživatele kontrolovat. Programátor je nucen si validace psát sám.


Aplikace Silverlightu lze validovat stejně dobře jako aplikace Flexu. Základní sada knihoven pro Silverlight validace neobsahuje, dá se však stáhnout externí validační knihovna, která obsahuje základní validace formulářových polí a drží se výše stanovených principů.


OpenLaszlo poskytuje validace formulářů ve stejné míře, jako Flex nebo Silverlight. Snad kromě regulárních výrazů a čísla kreditní lze validovat jak řetězce, tak čísla. Validátory jsou obsaženy v pomocných utilitních knihovnách a mohou sloužit jako šablona pro napsání validátoru vlastního.

Validace na straně serveru lze napsat pro libovolnou serverovou technologii, jako protikus klientské části aplikace. Avšak souvisí s tím režie síťových prostředků, zpracování logiky navíc a v nemalé míře zhoršená uživatelská přívětivost, pokud uživatel zadá a odešle data a aplikace mu následně po nějaké době oznámí, že data nejsou platná. Validace na straně klienta lze implementovat u všech RIA technologií. Z tohoto pohledu jsou aplikace zabezpečeny. Zda jsou validní data bezpečně přenášena mezi klientem a serverem popisuje následující srovnání.

### 4.2.5 Bezpečnost

Následující srovnání porovnává technologie ze dvou pohledů. Jak technologie řeší autentizaci uživatele a jak je zabezpečen přenos dat mezi klientem a serverem. U tohoto bodu je srovnání omezeno na podporu komunikace klient-server přes HTTPS.

ActionScript ani Flash platforma v základu nepodporují zabezpečenou klient-server komunikaci. Řešením je schovat výslednou Flex aplikaci za zabezpečené připojení, které zajišťuje prohlížeč a server. Tedy místo HTTP protokolu použít při načítání aplikace HTTPS. Omezení, které z toho plyne je, že aplikace načtená z HTTP nemůže posílat a získávat data z HTTPS.

Pro autentizaci uživatele může využít jeden z několika způsobů ověřování a to například LDAP (Lightweight Directory Access Protocol) nebo JAAS (Java Authentication and Authorization API).

Zajistit bezpečnost pro GWT aplikaci může být nelehký úkol. I když je aplikace většinou jednostránková, je však složena z HTML a JavaScriptu, což nahrává potencionálním útočníkům. Nejznámější z útoků může být XSS (Cross-Site Scripting), tedy útok právě pomocí skriptů vnořených do patřičné stránky. Pro GWT existuje několik doporučení a technik jak minimalizovat riziko napadení. Existuje také způsob jak použít SSL (Secure Sockets Layer) komunikaci a cookies (sušenky) pro autentizaci uživatele.

O zabezpečení JavaFX existuje omezené množství informací. Jelikož je však JavaFX postavená na Java platformě, může využít pro SSL komunikaci libovolný z dostupných Java bezpečnostních prostředků z rodiny Java SE Security.

Silverlight aplikace jsou plně přístupné přes HTTPS. I zde však existují jistá omezení a to například povolení pouze GET a POST metod, veškerá komunikace je asynchronní, nebo jen 202 OK a 404 Not Found kódy HTTP odpovědi jsou dostupné.

Pro autentizaci uživatele je potřeba využít a zkombinovat technologie z WCF (Windows Communication Foundation) a IIS (Internet Information Services), nebo využít možnosti použít ASP.NET FormsAuthentication.

OpenLaszlo aplikace stejně jako Flex aplikace běží ve Flash přehrávači a řídí se tedy stejnými principy při SSL komunikaci jako Flex aplikace.

Zpracování autentizace taktéž není pro OpenLaszlo aplikace problém. Realizuje ji pomocí cookies s jistým omezením. Jelikož běží aplikace ve Flash přehrávači, musí o cookie z webového prohlížeče někoho požádat. Jedním z řešení je požádat JSP stránku o její vrácení.

### 4.2.6 Sandbox

Již zmíněným prvkem RIA aplikací a také poslední zkoumaná funkční vlastnost je sandbox. Jedná se o prostředí s jasně definovanými pravidly a omezeními, ve kterém klient

RIA aplikace běží. Toto prostředí má za úkol stanovit jasná pravidla pro přístup k některým, většinou lokálním zdrojům, stejně jako omezit některé funkce a možnosti, které jsou vlastní desktopovým aplikacím a zamezit tak případnému zneužití klienta třetí stranou.

Úlohou sandboxu je také přidělení zdrojů klientovi, jako je paměť a dočasné místo na disku. Nejznámějšími sandboxy jsou applety, emulátory virtuálních strojů nebo pro OS Linux podpora jádra pro virtualizaci.

Flex RIA aplikace plně využívají prostředí Flash přehrávače. Ten jako takový poskytuje čtyři základní druhy sandboxu pro běh SWF aplikací. Tyto čtyři druhy poskytují omezení pro SWF aplikace stažené z Internetu, nebo je SWF aplikace lokální soubor buď s přístupem k Internetu, nebo k lokálním zdrojům. Poslední typ umožňuje spuštění aplikace lokálně s přístupem jak k Internetu, tak k lokálním zdrojům. Komunikace mezi jednotlivými sandboxy je vázána striktními pravidly.

Flash přehrávač se také v rámci omezení například stará o zpracování HTTPS nebo RTMP komunikace.

GWT aplikace sama o sobě neběží v žádném sandboxu. JavaScript je schován přímo v prohlížené stránce a tak se o omezení a zabezpečení stará webový prohlížeč. Z tohoto důvodu je GWT aplikace svázána s omezeními JavaScriptu. To může být nevýhoda pro psaní GWT aplikace, jelikož existuje větší riziko, že právě díky JavaScriptu bude aplikace napadena.

Omezení pro JavaFX aplikace lze rozdělit do dvou skupin. JavaFX aplikace lze spustit pomocí appletu nebo pomocí JWS technologie. Pro první přístup existuje sandbox, který se skládá ze tří prvků: bytekód verifikátor, který kontroluje, zdali je přeložený Java kód přeložen podle pravidel a lze jej spustit v JVM. Druhý kontrolní bod je classloader. Ten se stará o hladký průběh, tedy kdy a jak, načítaní tříd. Třetí složkou sandboxu je security manager, který hlídá používání veřejných rozhraní a sleduje operace, které applet provádí.

Druhý způsob spouštění JavaFX aplikací je omezen zejména systémem podepisování zdrojových kódů. Je tak zaručeno, že uživatel nespustí nic, co není označeno jako důvěryhodný obsah.

Zabezpečení Silverlight aplikace lze rozdělit do tří částí. Zabezpečení na úrovni aplikace, HTTP zabezpečení a zabezpečení na úrovni soketů. První část se stará o integraci Silverlight pluginu s webovým prohlížečem a zdali je dovoleno, aby Silverlight plugin volal DOM HTML strukturu webové stránky ze které je spouštěn. Druhá část byla zmíněna v kapitole 4.2.5 a třetí část řeší specifické omezení při využití komunikace pomocí soketů.

Pro OpenLaszlo aplikace generované do Flashe platí stejná omezení jako pro Flex aplikace. Díky tomu má OpenLaszlo aplikace například omezený přístup k lokál-

ním zdrojům. Pro aplikace generované do DHTML existují podobné doporučení, omezení a rizika jako pro GWT aplikace překládané do JavaScriptu.

Stejně jako u srovnání obecných vlastností, i na konci této kapitoly je uvedena pomyslná stupnice vítězů.


Vysvětlení, proč právě Flex, Silverlight a OpenLaszlo obsadily první tři příčky je zmíněno v části 4.4.2.

## 4.3 Zaměřeno na uživatele

Třetí a poslední srovnávací část je zaměřena na vlastnosti, které již nesouvisí ani s programováním ani vnitřním uspořádáním prvků RIA technologie, ale srovnává uživatelský dojem jednotlivých RIA klientů a jejich nasazení u uživatele.

Hodnocení look & feel, tedy vzhled UI a hodnocení webových prezentací jednotlivých technologií odráží subjektivní pohled na většinou bohaté grafické UI, které jednotlivé technologie nabízí. Srovnání pak doplňuje zmínka o možnosti distribuce RIA aplikací a jejich výkonnost.

### 4.3.1 Look & Feel

Flex nabízí mnoho UI komponent pro vytvoření opravdu líbivého obsahu. Výsledné Flash aplikace jsou proto většinou velmi zdařilé grafické prezentace, s větším množstvím animací, barevných kombinací, stínování a dalších.

Na druhou stranu jsou základní komponenty Flexu laděny do šedomodré neutrální barvy, při výběru či akci podbarvené světle modrou. Vytvořená aplikace je tak velmi líbivá a přehledná.

Odlišný přístup k UI byl, i díky technologiím použitým v základu GWT, zvolen právě u této technologie. Uživatelské rozhraní má základ nad Java knihovnou Swing a GWT aplikace vypadají do jisté míry podobně jako klasické Java desktop aplikace. Avšak díky mapování komponent na DOM strukturu jsou některé komponenty, například panel záložek, mírně odlišné a jinak zobrazené. V GWT aplikaci se tak nedá přehlédnout určitý konzervativní styl Java aplikací spojený s klasicky generovanou webovou prezentací. Na druhou stranu toto lze potlačit použitím CSS stylů a částečně tak dodat GWT klientovi kabát bohaté Internetové aplikace.

JavaFX již ve svém názvu obsahuje předzvěst, jak vypadá klient této technologie. Kompletní sada prvků je převzatá z knihovny Swing a pouze přidané možnosti

jako je rozšířená vektorová grafika a animace dělají z JavaFX klienta multigrafickou aplikaci. Pokud tedy JavaFX aplikace představuje zobrazení a editaci jednoduchých formulářových dat, bez většího zásahu grafika je JavaFX klient téměř k nerozeznání od Swing aplikace. Na druhou stranu lze vytvořit pomocí již zmíněných podpůrných metod bohatou prezentaci, která se může rovnat s prezentací Flashe

Klient Silverlight aplikace se dá složit z několika základních, nenápadných komponent. Překlad názvu Silverlight jako "Stříbrné světlo" přesně vystihuje základní cítění grafických komponent: komponenty jsou vesměs šedé s modrým okrajem či přechodem. Komponenty Silverlightu jsou tak na první pohled stylově velmi blízko právě Flex komponentám.

Pokud je však pro grafickou stránku Silverlight technologie použit pomocný nástroj pro generování bohatého UI, lze ze Silverlight prezentaci vytvořit k nerozeznání podobnou Flash prezentaci.

OpenLaszlo jde částečně svým vlastním směrem. Komponenty jsou laděny do šedivého tónu, design je velmi jednoduchý. Na první pohled tak může OpenLaszlo design připomínat X/Motif styl pro Linux. Rozdíl je pouze v zaoblení rohů dialogových oken, vroubkované horní lišty dialogových oken a přidání křížku do pravého horního rohu pro jejich zavření. Posledním rozdílem mezi X/Motif a OpenLaszlo stylem by mohlo být použití gradientu u některých prvků a možnosti využít šesti odlišných barevných stylů.

Základní sada prvků jednotlivých technologií jsou stylově a barevně jednotné komponenty, pomocí níž lze vytvořit jednoduchou prezentaci. Například přehled dat v tabulce, tedy výsledek zpracování dat z databáze. Zároveň lze pomocí různých metod doplnit či přestylovat sadu komponent tak, aby v aplikaci nebyl vidět ani jeden prvek v základní podobě a aplikace vypadala jako prezentace, například nakreslená v grafickém editoru. Tuto možnost jako jediná technologie nepodporuje GWT a to již díky zmíněné vlastnosti běhu v okně webového prohlížeče.

Dá se tedy říci, že aplikace, které ke svému běhu využívají externí runtime prostředí a nejsou přímo závislé na vykreslování prohlížečem, mají více volného prostoru při tvorbě UI. Za pivota těchto technologií lze považovat Flex. Flash aplikace jsou dnes již tak rozšířené, že se může zdát, jakoby existovaly dva světy: Flex technologie s Flash UI a zbytek. Přeci jen Flex obsahuje více komponent pro UI než Silverlight nebo JavaFX. Zůstává tedy otázka, zdali bude do budoucna vzorem pro vývoj komponent jednotlivých RIA technologií právě Flex.

### 4.3.2 Distribuce


Klient Flex aplikace se může k uživateli dostat dvěma způsoby: 1. přímo skrze Flash přehrávač s odkazem na SWF soubor, 2. pomocí technologie Adobe AIR,

která umožňuje běh Flex aplikace mimo webový prohlížeč. V prvním případě tedy uživatel přistupuje k aplikaci přes webové rozhraní a výsledek vidí na určitém místě jako Flash prezentaci.


Aplikace GWT je reprezentována webovou stránkou s určitým množstvím JavaScriptu. Spuštění aplikace tedy vyžaduje pouze zadání konkrétní adresy do prohlížeče.


Více možností pro spuštění aplikace poskytuje také JavaFX. Standardně se aplikace spustí v appletu. Je však možno ji také z tohoto appletu "vytáhnout" a používat mimo prohlížeč. Dalšími možnostmi spuštění jsou JWS a emulace pro mobilní zařízení.


Silverlight aplikace je spuštěna v pluginu pomocí odkazu na archiv aplikace. Stejně jako u Flexu je tedy spuštění provedeno pomocí prohlížení webové stránky.


Pro spuštění OpenLaszlo aplikace musí na ni uživatel znát adresu. Při jejím zadávání lze pak dále určit, zdali bude aplikace vrácena ve Flashi či DHTML. Do webového prohlížeče je pak natažen zkompilovaný kód klienta.

Distribuce RIA aplikace, myšleno její klientské části, se děje ve většině případů pomocí webového prohlížeče.

Klient však musí být někde uložen. Většinou je to někde na serveru, spolu s dalšími pomocnými soubory. Distribuci celé aplikace lze tak provést zabalením klientské i serverové části do WAR archivu a tento pak nahrát na příslušný webový server. Kromě OpenLaszlo technologie, kde se klientská část kompiluje až při prvním volání aplikace, si jednotlivé RIA technologie z tohoto web serveru klienta stáhnou a spustí.

RIA aplikace lze tedy distribuovat jako WAR archivy, které obsahují minimálně klientskou část spolu s externími soubory jako jsou obrázky či multimédia.

### 4.3.3 Interakce uživatele při instalaci

Komfort a pohodlí při instalaci zajišťuje u desktopových aplikací instalátor. Tedy program, který pomocí několika kroků nainstaluje celý program bez toho, aniž by uživatel potřeboval znát cesty, kam se nahrávají potřebné zdrojové soubory. U RIA aplikací je tomu jinak. Jelikož jsou nahrávány přímo do prostředí webového prohlížeče, stará se o instalaci (natažení aplikace) samotný prohlížeč. Pouze pro technologie, které jsou spouštěny v runtime prostředí je nutná interakce uživatele. Aktualizace aplikací, jak bylo řečeno dříve je u RIA aplikací prováděna automaticky s dalším spuštěním. Pouze aktualizace pluginů je jednou za čas nevyhnutelná.


Uživateli stačí zadat příslušnou webovou stránku a pokud má konkrétní webový prohlížeč nainstalován Flash přehrávač, aplikace je přímo nahrána. Uživatel za-

sáhne pouze v případě, že je potřeba Flash přehrávač stáhnout či aktualizovat. Jiná akce od uživatele není vyžadována.

 GWT aplikaci uživatel spustí pokud si zobrazí příslušnou webovou stránku. Pokud má však v prohlížeči nastaveno vysoké zabezpečení, nebo jednoduše vypnut JavaScript, musí tento zapnout. To je jediná možná akce uživatele při spouštění GWT aplikace.

 Pokud je JavaFX aplikace spouštěna přes JWS a pokud nemá uživatel nainstalovanou Javu, musí ji doinstalovat. Tato možnost je mu nabídnuta automaticky a instalaci Javy provede instalátor. Po spuštění a při stahování pak pouze uživatel povolí či zakáže spuštění aplikace. Tento krok je přidán z důvodu zabezpečení. Java i JWS se aktualizují automaticky, pokud to uživatel nezakáže.

 Stejně jako u Flexu je uživatel vyzván pouze ke stažení či aktualizaci Silverlight pluginu. Žádná další akce od uživatele není vyžadována.

 Uživatel OpenLaszlo aplikace bude zasahovat, podobně jako u Flexu nebo GWT, pouze v případě, pokud nebude mít nainstalován Flash přehrávač, nebo bude mít zakázán JavaScript. A to v případech spuštění aplikace jako Flash prezentace nebo v DHTML.

### 4.3.4 Výkonnost

Měření výkonnosti může být provedeno několika způsoby a zpravidla obsahuje testy z více oblastí dané technologie. V následujícím srovnání jsou uvedeny postřehy výkonnosti jednotlivých RIA technologií, především však z oblasti vývoje a rychlosti UI. Přeci jen jsou RIA určeny pro běh na klientském prostředí a využívají pro zobrazení obsahu prostředky klienta.

 Při větším prostorovém výpočtu může být zátěž na klientské prostředky veliká. Je proto doporučeno věnovat přiměřenou část právě návrhu počtu, rozvržení a vzájemného propojení jednotlivých UI komponent.

Díky stažení kompletní aplikace na klienta je zátěž na síťové prostředky menší, než kdyby byla stejná aplikace psána například pomocí JSP technologie. Je to dáno hlavně tím, že Flex aplikace je jednou na serveru zkompilována a natažena na klienta. U JSP je nejdříve připravena úvodní stránka a s každým dalším voláním serveru musí server vygenerovat nový obsah.

 GWT aplikace, která je zobrazována přímo v prohlížeči může obsahovat několik slabých míst a to jak při startu tak při výkonu. Existuje však několik rad pro psaní

optimálního klientského kódu. Jsou to například: komprese zdrojových kódů, cache aplikace na klientovi a možnost balení obrázků nebo CSS stylů do jednoho balíčku, což zlepší start aplikace nahrazením více volání na zdrojové soubory pouze jedním.


Díky tomu, že je JavaFX aplikace zabalena a distribuována přímo na klienta a v jednom požadavku, může být spuštění aplikace na klienta stejně rychlé jako u Flexu nebo Silverlightu. JVM je optimalizováno pro interpretaci přeloženého bytekódu, avšak opět stejně jako u Flexu či Silverlightu více výpočtů nad 2D grafickými komponentami může aplikaci výrazně zpomalit.


Při optimalizaci výkonnosti Silverlight aplikací by měl být kladen důraz na uvážené rozmístění a použití grafických komponent, stejně tak jako jejich počet a logiky, které představují. Největší možnou výkonnostní trhlinou Silverlight aplikací se tak stávají prostorové transformace, animace či optimalizace přehrávání multimédií.


Pro OpenLaszlo aplikaci existuje, stejně jako pro ostatní RIA technologie, několik doporučení pro optimalizaci spouštění aplikace či její výkonosti při běhu. Může jít o kompresi aplikace kompilované do DHTML, efektivní správu paměti, vnořené třídy nebo některé metody pro ruční optimalizaci přiřazování atributů.

### 4.3.5 Dostupnost aplikací - indexování, vyhledávání

Vyhledávání dat v Internetu je nelehký úkol, který ještě ztíží datové formáty jednotlivých RIA aplikací. Flash prezentace nemusí obsahovat mnoho textu, OpenLaszlo aplikace může být kompilována jak pro Flash tak pro DHTML, JavaFX aplikace běží v prostředí Java appletu, Silverlight běží také ve vlastním runtime prostředí. V této kapitole je tedy uvedeno několik základních postřehů k vyhledávání RIA aplikací a SEO (Search Engine Optimization), tedy úpravy Internetových aplikací pro vyhledávací stroje (vyhledávací engine).


Nejznámější vyhledávače současnosti Google a Yahoo již podporují vyhledávání Flash aplikací. Tedy stačí zadat při vyhledávání rozšiřující požadavek na vyhledávání právě v SWF obsahu.


GWT a SEO nejdou ruku v ruce. Ve výsledné GWT stránce totiž nemusí být ani náznak, o jakou prezentaci či aplikaci se jedná. Vše je ukryto v přiloženém JavaScriptovém souboru a vyhledávací engine tedy nemá možnost webovou stránku prohledat.


JavaFX je na tom ještě hůře. Klientská část aplikace běží v prostředí Java appletů, tedy kompletně v prostředí, které vyhledávací stroje nejsou schopny prohledat. Pokud je aplikace spouštěna přes JWS, je šance nalézt aplikaci téměř nulová.

Pro SEO Silverlight aplikací existuje několik způsobů. Jedním z nich je například XSLT (eXtensible Stylesheet Language Transformations) transformace XAML kódu do XHTML (Extensible Hypertext Markup Language). Pro vyhledávací engine se tak obsah Silverlight aplikace stane viditelný a pokud je na klientovi nainstalován Silverlight plugin, dokáže XHTML kód zobrazit jako Silverlight aplikaci. Jednou z dalších možností je rozvrstvení Silverlight aplikace na viditelnou a neviditelnou část, kde je do neviditelné části ukryt obsah pro vyhledávací stroje.

Pro vyhledávání OpenLaszlo aplikací existuje, pokud je výstup generován jako Flash aplikace, stejná možnost jako u Flex technologie. Pokud je ovšem aplikace generována do DHTML, existuje riziko, že použité dynamické prvky jako je JavaScript nebudou vyhledávači zpracovány. V tomto případě je nutno již při návrhu a vývoji aplikace dbát na SEO zásady.

Pro většinu technologií je dále doporučeno použít buď zabalení celé aplikace do webové stránky, která obsahuje prvky snadno rozpoznatelné vyhledávacími stroji, nebo použít seznam odkazů jako v podobě jednoduché HTML stránky, kde každý odkaz ukazuje na konkrétní část či celou RIA aplikaci.

### 4.3.6 Prezentace vydavatele

Na závěr srovnávací části této práce, je uveden seznam společností, které stojí za jednotlivými technologiemi.

Flex je technologie firmy Adobe Systems, jako jedna z jeho mnoha technologií a navazuje na úspěch Flash platformy a ActionScriptu.

Webové prezentace Flex technologie jsou na dobré obsahové úrovni, avšak chybí jim ucelená forma. Jednotlivé informace uživatel musí hledat na více místech.

Google Web Toolkit představuje rozšíření základny Google produktů jako jsou například Google Maps nebo Picasa.

Prezentace technologie je na dobré obsahové i formální stránce. Vše je na jednom místě v dobře čitelné podobě.

Technologií JavaFX se společnost Sun Microsystems snaží také uchytit na poli RIA.

Pro potencionálního zájemce o JavaFX technologii však budou na prezentačních stránkách nejspíše chybět některé detailnější technické informace. Prezentace je zaměřena spíše na prezentaci technologie v širším měřítku s několika hotovými, avšak poměrně komplexními aplikacemi, které nelze “rychle” naprogramovat.

Silverlight je jednou z mnoha technologií a produktů firmy Microsoft. Zároveň rozšiřuje platformu .NET o důležitou část, kterou se vývoj RIA aplikací v dnešní době stal.

Informace o technologii jsou dostupné na jednom místě, v relativně přehledné, graficky, formálně i obsahově dobré podobě.

Společnost Laszlo Systems nabízí technologii OpenLaszlo jako jeden ze svých dvou produktů.

Webová prezentace společnosti i technologie jsou na velmi pěkné grafické úrovni. Vše je řešeno přehledně pomocí jednoduchého a vkusného designu.

Většina aplikací napsaných v jednotlivých technologiích a dostupných na Internetu je zdarma, nebo minimálně jako ukázková aplikace. Díky tomu, že jsou RIA technologie zatím v relativně raném stádiu vývoje, jsou volně dostupné aplikace psány lidmi, kteří se hlouběji zabývají vývojem webových, nebo RIA aplikací.

Většina z výše uvedených společností však dodává také software napsaný v konkrétní RIA technologii jako komerční produkt buď jiným velkým společnostem, nebo technologii používá pro svou vlastní prezentaci.

Na závěr je uveden opět pomyslný stupínek vítězů.


Bodové hodnocení třetí srovnávací části je na prvních třech místech velmi vyrovnané díky všeobecně dobrým uživatelským vlastnostem. Technologie OpenLaszlo a GWT pak ztrácí hlavně díky horší možnosti vyhledání aplikace na Internetu.

## 4.4 Výhody, nevýhody RIA technologií

V této části jsou u jednotlivých technologií vyzdviženy jejich silné, ale i slabé stránky. Vodítkem pro toto hodnocení je obsah kapitoly 4, ve které jsou technologie srovnány podle typických vlastností RIA aplikací.

### 4.4.1 První část - obecné vlastnosti

Mezi hlavní kladné obecné vlastnosti Flexu patří zejména bohatý obsah SDK a grafický editor obsažený ve Flex Builder IDE. Díky spojení těchto dvou vlastností vzniká pro programátora ideální základ pro vývoj komplexní aplikace.

Naopak jedna z hlavních záporných vlastností je, že výše zmíněné IDE není zdarma. Další drobnou nevýhodou může být čas strávený studiem, který je potřeba před vývojem první aplikace.

GWT je vhodné pro programátory, kteří přechází z technologií jako je webový framework Apache Struts, technologie JSP, nebo čisté HTML programování. Použitý jazyk Java spolu s jednoduchým vytvářením GUI, stejně jako jednoduchý způsob implementace komunikace klient-server se odráží v nízkém čase potřebném k naučení GWT technologie.

Technologii lze z obecných vlastností vytknout snad jen menší dostupnost informací, které nejsou přímo uvedeny v dokumentaci.

Hlavní kladnou vlastností JavaFX technologie je stejně jako Flexu bohaté IDE NetBeans. Výhodou oproti Flexu je, že toto IDE je zdarma a vývoj JavaFX aplikace je tak dostupné pro větší okruh lidí.

Nevýhodou JavaFX technologie je bezesporu ovládnutí JavaFX Script syntaxe. Obecně čas potřebný k seznámení s technologií a proniknutí do vývoje komplexnější aplikace se řadí u srovnávaných technologií mezi nejdelší.

Při vývoji Silverlight aplikace programátor ocení bohatou sadu knihoven, která je podporována širokým výběrem programovacího jazyka pro psaní logiky klienta. Rovněž bohatá a přehledná dokumentace s návazností na ostatní Microsoft technologie činí ze Silverlightu nejlépe dokumentovanou RIA technologií.

Hlavní nevýhoda při začínání se Silverlight technologií je obsah SDK. Kromě knihoven neobsahuje téměř nic a pro vývoj aplikace je tedy nutné instalovat IDE. Jedním ze záporných bodů se může zdát relativně delší čas na naučení základních programovacích postupů.

Technologie OpenLaszlo, jako druhá nejlépe hodnocená technologie první části, vyniká především veřejnými zdrojovými kódy všech částí technologie, bohatou, avšak přehlednou dokumentací a obsahem SDK.

Jako nevýhodu lze uvést čas, který je potřeba pro seznámení s technologií, zejména syntaxí LZX skriptu.

### 4.4.2 Druhá část - funkční vlastnosti

Oproti obecným vlastnostem, kterým kraloval GWT, se již u funkčních vlastností projevují rysy RIA aplikací (kapitola 3.1). To se projevuje nejen na bodovém hodnocení srovnání, ale také na postupném převažování bohatých technologií Flex, Silverlight a OpenLaszlo.

Kladných vlastností za funkční část existuje pro Flex více. Největší výhodou je však bezesporu zásoba grafických komponent pro tvorbu bohatého GUI. Další pozitivní vlastností je možnost načítání dat a to jak z DB, serveru, či jiného datového zdroje.

Nevýhodou je naopak užší podpora multimediálních, tedy audio a video formátů, stejně jako nutnost použití externího nástroje pro vývoj 3D aplikace.

Předností GWT jsou hlavně v možnosti propojit klienta s libovolným zdrojem dat. V návaznosti na připojení ke zdroji dat lze vyzdvihnout obsah knihoven, které podporují jak zpracování zdrojových dat, tak komunikaci mezi klientem a serverem pro jejich získání.

Slabé místo GWT je však oblast multimédií a grafiky. GWT je spíše přizpůsobeno pro zpracování databázových dat a formulářů, než pro vytvoření prezentace bohaté na animace či dokonce video. Za další možný záporný bod je možno považovat absenci sandboxu, jako schránky pro klientskou část aplikace, chránící ji před možným zneužitím.

U JavaFX technologie lze nejvíce ocenit podporu pro tvorbu grafických prezentací i přes absenci generování 3D prvků. Stejně tak možnost propojit JavaFX aplikaci s více multimediálními formáty je pozitivní vlastnost JavaFX technologie.

Kde však JavaFX ztrácí na své konkurenty je zabezpečení komunikace klient-server a neexistence podpory validací. Reputaci částečně napravuje Java sandbox a bezpečnostní opatření při spouštění samotné JavaFX aplikace.

Silverlight technologie vyniká ve všech směrech. Za hlavní kladné vlastnosti lze označit bezpečnost, bohatou sadu komponent, nebo širokou podporu pro načítání dat.

Nevýhodou u Silverlight technologie může být omezené množství podporovaných multimediálních formátů.

OpenLaszlo obsahuje prostředky jak pro bezpečnou komunikaci klient-server, tak pro validaci dat. Silnou stránkou je také podpora multimediálních formátů a podpora pro animace ve 2D grafice.

Jako největší nevýhodu OpenLaszlo technologie lze označit absenci 3D a jako u všech technologií, absence přímého přístupu do databáze.

### 4.4.3 Třetí část - uživatelský zážitek

Každý, kdo si spustí aplikaci napsanou v jedné ze srovnávacích technologií bude mít pocit, že se jedná o aplikaci desktopovou v prohlížeči. Flex nebo JavaFX technologie k tomu přidávají možnost vytvořit navíc aplikaci bohatou na animace, obrázky a různé grafické vylepšení. Z tohoto pohledu jsou srovnávané RIA technologie hodnoceny kladně.

Flex, jak bylo řečeno výše, přidává aplikacím v Internetu bohatost, tedy aplikace vypadá designově velmi zdařile. Pokud je do aplikace přidán multimediální obsah, stane se z aplikace bohatá grafická prezentace.

Pozitivní je možnost vyhledat Flex aplikace v nejrozšířenějších internetových vyhledávačích Google a Yahoo, které již dokáží vyhledávat a procházet soubory formátu SWF.

GWT zaujme klasickým designem webových aplikací a kaskádových stylů. Výhodou GWT aplikace je absence instalace pomocného pluginu.

Díky omezené podpoře vyhledávání v Internetu však může být viditelnost GWT aplikací menší, než konkurenční Flex nebo OpenLaszlo aplikace.

Předností JavaFX aplikace je určitě klasický design aplikací napsaných v knihovně Swing, rozšířený o některé stylové grafické prvky. Vzhled aplikace tak může konkurovat prezentacím Silverlightu nebo Flexu.

Nevýhodou, stejně jako u GWT, je nízká podpora indexovatelnosti, tedy malá viditelnost pro vyhledávací stroje. Tedy menší pravděpodobnost, že uživatel hledající konkrétní aplikaci narazí na verzi psanou v JavaFX technologii.

Hlavní výhodou Silverlight technologie je možnost vytvářet aplikace, které jsou bohaté na grafické prvky a které mohou konkurovat aplikacím ve Flashi.

Za mírné komplikace při spouštění aplikace lze považovat snad jen nutnost zásahu uživatele při instalaci Silverlight pluginu, nebo opět omezenou možnost vyhledávání aplikace v Internetu.

OpenLaszlo aplikace může těžit především z vkusného designu grafických komponent, které lze renderovat v několika odstínech. Design aplikace je tak jednoduchý a připomíná desktopové aplikace postavené nad množstvím oken a formulářů. To nemusí být vůbec na škodu právě, například při potřebě vytvořit aplikaci s více formuláři pro zpracování dat.

Uživatelskou přívětivost může kazit snad jen delší doba při spouštění aplikace, která je způsobená kompilací při spuštění. Stejně jako u ostatních technologií je dále nevýhodou vyhledávání v Internetu. Řešení vyhledávání pomocí SWF formátu souboru, do kterého je aplikace kompilována musí být podmíněno dostupností zkompilované verze aplikace, což nemusí být vždy splněno.

# 5 Praktické využití RIA technologií

Technologie Flex obecně v základu obsahuje vše, co je k vývoji RIA aplikace potřeba. Flash je široce rozšířeným a známým prvkem Internetu a uživatelé jsou již s tímto prvkem seznámeni. Flex je tedy vhodným kandidátem jak pro seznámení s RIA technologiemi obecně, tak pro vývoj komplexní RIA aplikace. GWT je ideální pro začínající RIA programátory se základní znalostí jazyka Java a HTML. Pro JavaFX technologii se uživatel nejspíše rozhodne, pokud bude chtít využít stabilního zázemí JRE, tedy všech výhod, které vývoj nad Java technologiemi přináší. Silverlight přináší do světa RIA možnost tvorby bohatých prezentací spolu s napojením na technologie .NET platformy. Proto po této technologii nejspíše sáhnou programátoři zběhlí v právě v jazycích .NET platformy a XML. Konečně OpenLaszlo může sloužit buď jako technologie pro tvorbu profesionálních Internetových aplikací, či jako technologie která může sloužit jako testovací a na které se může programátor seznámit se všemi prvky a postupy vývoje RIA aplikací.

## 5.1 Kritéria pro výběr technologie

Proces výběru technologie pro jakýkoliv softwarový projekt není jednoduchá záležitost. Softwaroví analytici, produktoví manažeři a obchodníci musí vzít v úvahu mnoho aspektů vývoje v dané technologii, stejně tak jako její schopnosti prosadit se na trhu. Tento proces může trvat delší dobu a nemusí být vždy úspěšný. Respektive může být na počátku vývoje zvolena nevhodná technologie a toto rozhodnutí se poté promítne do celého procesu vývoje softwarového díla, které také nemusí být díky špatnému rozhodnutí vůbec dokončeno.

Srovnání v kapitole 4 poskytuje základní představu o tom, co která technologie nabízí a jaké jsou její hlavní kladné a záporné vlastnosti. V této kapitole je popsán jednoduchý postup, jak zvolit technologii podle konkrétních potřeb a požadavků na výsledné softwarové dílo, ať už jednoduchou či bohatou Internetovou aplikaci.

**Krok 1**

RIA technologie jsou určeny především k tvorbě aplikací, které svými možnostmi převyšují klasické statické webové prezentace. Základní otázkou může tedy být:

Je potřeba aplikaci s jednoduchým uživatelským prostředím, tedy bez nutnosti větší interakce aplikace s uživatelem?

- Pokud **ano**, pak je nejlépe uvažovat o použití klasického programování HTML stránek, s jednoduchou strukturou a standardním rozložením grafických prvků. Stejně tak bude výsledná aplikace dostupná komukoliv, jednoduše skrze webový prohlížeč.
- Pokud **ne**, je čas podívat se blíže na možnosti jednotlivých RIA technologií.

Toto základní členění může již na počátku procesu výběru rozhodnout, kterým směrem se bude vývoj aplikace ubírat. Pro zvolení RIA technologií většinou rozhodnou

požadavky na aplikaci jako jsou vzhled podobný desktopovým aplikacím, tedy použití tlačítek, menu, panelů a dialogových oken pro interakci aplikace s uživatelem. Dále především plynulý chod aplikace a dynamické načítání jednotlivých částí aplikace.

**Krok 2**

Po provedení prvního výběru lze dále specifikovat požadavky na aplikaci a rozdělit tak RIA technologie do tří kategorií. Tyto kategorie odráží rozdělení RIA technologií podle obrázku 1 v kapitole 3. Na obrázku jsou RIA technologie rozděleny do tří částí podle jednotlivých otázek:

- RIA postavené nad AJAXem – bude aplikace dostupná na téměř všech webových prohlížečích a všem uživatelům Internetu?
- Java technologie – bude potřeba, aby technologie splňovala kritéria jako je bezpečnost, znovupoužitelnost, rozšířitelnost, výkonnost?
- Flash – bude potřeba vytvořit bohatou, designově propracovanou a na grafické prvky bohatou aplikaci?

Ačkoliv je v tomto kroku již vybrána oblast RIA technologií, nemusí konkrétní kategorie splňovat všechna kritéria, která byla stanovena v prvním kroku rozhodování mezi HTML a RIA technologiemi.

**Krok 3**

K jednotlivým kategoriím lze sestavit jednoduchý seznam dalších požadavků, podle kterých by měla být kategorie označena za přijatelnou a tedy vhodnou pro výběr konkrétní RIA technologie.

- RIA postavené nad AJAXem – tvorba aplikací psaním čistého HTML s použitím AJAXu bez použití podpůrných frameworků je velmi obtížná. Programátor se musí vyrovnat s různou implementací pro různé prohlížeče a OS. Další požadavky kladené na tuto kategorii jsou ze strany tak vychvalované funkcionality RIA aplikací jako jsou UI prvky a uživatelský zážitek. To znamená, jakým způsobem lze zobrazit uživateli data a případně jak náročné bude vyvinout bohaté UI pokud ho technologie nebude poskytovat sama. Další otázky, například bezpečnosti klient-server komunikace nebo podpory dalších programovacích jazyků a technologií, mohou zpřesnit správné rozhodnutí.
- Java technologie – tyto technologie mají obecně dobrý a pevný základ v široké, technicky vyspělé platformě. RIA technologie založené nad Javou těží z podpory bohatých grafických knihoven, možnosti běhu pro mobilní a jiné zařízení, standardizovaném vývoji rozsáhlých Java aplikací a podpory vývoje v podobě programovacích nástrojů a prostředí.

Pokud se bude rozhodovat nad technologiemi této kategorie, tak mohou být předneseny konkrétní požadavky a dotazy právě z oblastí vývoje UI, komunikace klient-server nebo podpory běhu aplikace v různých prostředích. Java a její technologie se relativně rychle vyvíjí. Otázka aktualizace runtime prostředí by měla také hrát svou roli v rozhodovacím procesu.

- Flash – Všechny vlastnosti technologie Flash zastíní opravdu bohatý obsah spojený s bohatým uživatelským zážitkem. Pokud padne výběr na RIA technologii z této kategorie, jsou otázky ohledně podpory a vývoje vlastních UI prvků na místě. Stejně tak je potřeba zvážit použití runtime prostředí v podobě Flash přehrávače a jeho podpoře webovými prohlížeči.

  Dalším kritériem pro aplikace psané pro Flash platformu může být skloubení ActionScriptu s jinými technologiemi. Pokud totiž bude například pro projekt dostačující bohatá sada UI prvků a podpora animací avšak pro napsání servisní vrstvy či propojení se serverem nebude k dispozici potřebná technologie, stane se v té chvíli Flash nepoužitelným i přesto, že bude nejvhodnější kandidát pro napsání klienta. Stejně tak bude důležitá otázka zajištění bezpečnosti.

**Krok 4 - výběr konkrétní technologie**

Na závěr je uvedena možná varianta výběru při rozhodování použití jednotlivých technologií:

- RIA postavené nad AJAXem - do této kategorie lze zařadit jednoznačně technologii GWT.
- Java technologie - jediný zástupce technologie postavené čistě nad Java platformou je JavaFX.
- Flash - není překvapením, že hlavním kandidátem pro vývoj aplikací z této kategorie je Flex. Bohaté aplikace lze však vytvořit také v technologii Silverlight.

Zdá se, že technologie OpenLaszlo nezapadá přímo do žádné kategorie. Je to dáno tím, že spojuje jak bohatý obsah, avšak ne tak bohatý jako u Flexu nebo Silverlightu. Technologie není postavena ani nad jazykem Java a prvky jako je výkonnost, rozšířitelnost či bezpečnost sice splňuje, ale pro jejich dosažení je potřeba již značná zručnost a zkušenost. Možnost DHTML výstupu zařazuje technologii také spíše do AJAX kategorie, i když výstup bude generován nejčastěji jako Flash aplikace. OpenLaszlo tak zůstává někde mezi jednotlivými technologiemi a jeho volba je závislá na podrobnějším prozkoumání jeho konkrétních možností.

## 5.2 Ilustrační příklady vybraných technologií

Pro prezentaci jednotlivých technologií byly vybrány tři základní vlastnosti:

- Vytvoření grafického dialogového okna.
- Základní komunikace klient-server.
- Načtení a zpracování multimediálního obsahu - v ilustračních aplikacích se jedná o přehrávání videa.

Při všech implementacích byly použity základní technologické postupy, při komunikaci klient-server se jednalo o nejčastěji použitý způsob získávání dat ze serveru. Jako programovací nástroj byly použity Eclipse IDE, NetBeans IDE, Flex Builder IDE a MVS. Výsledné aplikace se podařilo zabalit do WAR archivů, takže se dají distribuovat jako webové aplikace. Vývojovým prohlížečem byl Mozilla Firefox 3.0 a Internet Explorer 7.

### 5.2.1 Spuštění ilustračních příkladů

Ilustrační příklady jsou distribuovány jako WAR archivy. Lze je tedy jednoduše nahrát na webový server a spustit. Z důvodu vývoje v lokálním prostředí a pouze základní implementaci klient-server komunikace a přehrávání videa, jsou některé odkazy na datové zdroje a Webové služby odkazovány přes lokální URL a standardní HTTP port. Pro korektní běh aplikací je tedy potřeba mít nastaven kontext webového serveru na *http://localhost:8080*. Prezentace technologie JavaFX navíc potřebuje ke svému běhu JRE. Ukázková aplikace byla vytvořena nad poslední verzí JRE a pro korektní spuštění je tedy potřeba mít nainstalovánu jak podporu v prohlížeči, tak spuštění webového serveru nad danou verzí. Poslední verze JRE je 1.6.0_13.

Konkrétní příklady spuštění jsou v následujícím seznamu:


Ukázková aplikace je složena ze dvou částí: Klienta a Webové služby, která je však shodná s implementací pro Silverlight aplikaci. WAR archiv klientské části je složen jak ze samotného MXML zdrojového kódu, tak již zkompilované SWF verze aplikace. Archiv lze automaticky vytvořit v Flex Builder IDE a programátor se tak nemusí starat například o doplnění pomocných skriptů pro volání Webové služby.

Po nahrání na server je aplikace dostupná na adrese:

*http://localhost:8080/PrezentaceFlex*

GWT archiv lze vytvořit ručně, avšak Eclipse IDE a předpřipravené buildovací skripty pro vývoj GWT aplikace nahrají všechny potřebné skripty na jedno místo. Tyto soubory spolu se zkompilovanými zdrojovými kódy RPC objektů je tak možno snadno přenést na jedno místo a vytvořit z nich WAR archiv. Ukázkový příklad komunikuje pomocí RPC a není tedy třeba žádné Webové služby. Aplikace je dostupná na adrese:

*http://localhost:8080/PrezentaceGWT*


Podobně jak o u GWT, JavaFX archiv je potřeba vytvořit ručně a to složením vygenerovaných JAR archivů a JNLP deskriptorů. Ty však NetBeans IDE generuje také automaticky při buildu aplikace a je tedy potřeba je přenést na jedno místo a zabalit. Při balení je nutno přejmenovat URL serveru v příslušných odkazech, jelikož při vývoji používá NetBeans IDE svůj vlastní server. Automaticky lze zabalit Webovou službu vytvořenou přímo v NetBeans IDE. Tyto dva archivy po nahrání na server zpřístupní JavaFX prezentaci na adrese:

*http://localhost:8080/PrezentaceJavaFX*


Vývoj Silverlight aplikace v Eclipse IDE přináší úskalí v podobě ručního vytváření distribučních archivů. V tomto případě je nutné vytvořit jak archiv s aplikací, tak archiv s Webovou službou. Všechny potřebné soubory lze však získat z adresářů projektu. Archiv s aplikací obsahuje pouze externí datové zdroje, což je v ukázkovém příkladě video, XAP archiv se Silverlight aplikací, startovací stránku a pomocný skript pro spuštění aplikace. Archiv Webové služby obsahuje její deskriptor, zkompilované zdrojové kódy implementace Webové služby a další pomocné kódy pro spuštění a přístup k Webové službě. Aplikace je po nahrání těchto dvou WAR archivů na server dostupná na adrese:

*http://localhost:8080/PrezentaceSilverlight*


Největší ze všech arhivů je WAR archiv pro OpenLaszlo aplikaci. V tomto archivu je obsažen kompletní LPS spolu se zdrojovými kódy knihovních komponent a nástroji pro překlad. Je to proto, že při distribuci OpenLaszlo aplikace není možno spoléhat na to, že bude aplikace nahrána právě pod LPS. Díky tomuto opatření tak lze spustit OpenLaszlo aplikaci z libovolného webového serveru. Ten bude mít spolu se zdrojovým kódem aplikace dostupnou i implementaci LPS a bude se tak tvářit jako originál. Kromě LPS je v archivu již jen zdrojový kód aplikace. WAR archiv lze zabalit v Eclipse IDE kompletně. URL aplikace je mírně odlišné od předchozích příkladů a to:

*http://localhost:8080/PrezentaceOpenLaszlo/prezentace.lzx*

Posledním WAR archivem, který doplňuje ilustrační příklady, jsou videa. Jedná se pouze o jednoduchou webovou aplikaci, kerá obsahuje videa pro připravené prezentace technologií OpenLaszlo a JavaFX. Zbylá dvě, pro Flex a Silverlight, jsou uložena v příslušných archivech.

### 5.2.2 Grafická komponenta Hello World

Dialogové okno s nápisem Hello World slouží pro demonstraci, jak by mohlo vypadat ve zvolené technologii chybové či jiné hlášení. U většiny technologií je vytvoření dialogového okna velmi intuitivní. Některé technologie však neobsahují přímou podporu dialogového okna a programátor si tak musí dialogové okno doslova “složit” sám.

```
<mx:TitleWindow y="250" width="250" height="200" layout="absolute"
    visible="false" title="Okno s HelloWorld" id="oknoHelloWorld" x="79"
    horizontalAlign="left">
 <mx:Label text="Hello World!!" fontWeight="bold" fontSize="24" width="195"
    height="37" x="4" y="4"/>
</mx:TitleWindow>
```

Výpis 1: Flex - Dialogové okno s Hello World

Zdrojový kód pro vytvoření dialogového okna je v MXML velmi jednoduchý. Flex obsahuje přímo komponentu pro vytvoření modálního okna, které stačí pouze nastavit příslušné parametry a obsah. Grafická reprezentace tohoto zdrojového kódu je na obrázku 8. Dialogové okno je zobrazeno uprostřed šedomodré plochy, která představuje pozadí celé Flex aplikace.


Obrázek 8: Flex - Dialogové okno s Hello World

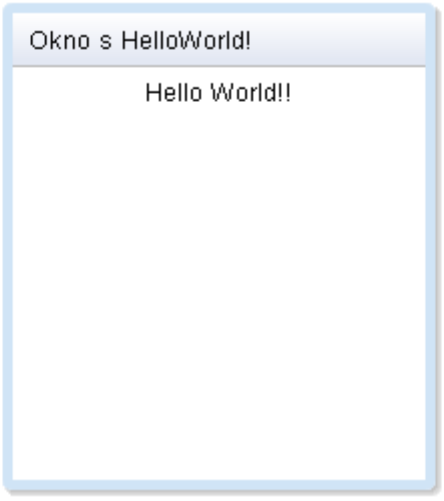
Vytvoření dialogu v GWT je nápadně podobné programování v knihovně Swing. I zde je vytvořena jednoduchá komponenta, které jsou nastaveny základní vlastnosti umístění a přidán text *Hello World*. Ten je v komponentě reprezentován konstantou. Díky podpoře lokalizace lze jednoduše uložit a načítat požadované řetězce v různých jazycích.

```
DialogBox dialogBox = new DialogBox(false, false);
dialogBox.setText(MESSAGES.titulekOkna());
dialogBox.setAnimationEnabled(true);

VerticalPanel dialogVPanel = new VerticalPanel();
dialogVPanel.setWidth("200");
dialogVPanel.setHeight("200");
dialogVPanel.setHorizontalAlignment(VerticalPanel.ALIGN_CENTER);
dialogVPanel.add(new Label(MESSAGES.helloWorld()));

dialogBox.setWidget(dialogVPanel);
dialogBox.setPopupPosition(150, 150);
```

Výpis 2: GWT - Dialogové okno s Hello World




Obrázek 9: GWT - Dialogové okno s Hello World

Výsledná grafická podoba je na obrázku 9 jako jednoduché dialogové okno s nápisem *Hello World!!*, tentokrát bez barevného pozadí díky standardnímu bílému pozadí GWT aplikace.

Vytvoření dialogového okna v JavaFX Scriptu nelze udělat přímočaře. Je třeba vzít komponentu okna z knihovny Swing a přidat ji do JavaFX aplikace.

```
var oknoHelloWorld = new JInternalFrame();
var jPanel = new JPanel();

...

jPanel.add(new JLabel(##"helloWorld"));
oknoHelloWorld.add(jPanel);
oknoHelloWorld.setPreferredSize(new Dimension(200, 200));
oknoHelloWorld.setTitle(##"titulekOknaHelloWorld");

...

SwingComponent.wrap(oknoHelloWorld)
```

Výpis 3: JavaFX - Dialogové okno s Hello World

V první části jsou vytvořeny příslušné komponenty. V druhé části je jim nastavena velikost a obsah. Třetí část je zabalení komponenty z knihovny Swing do komponenty, kterou již lze přeložit JavaFX Script překladačem. Pro dosažení stejného designu jako mají ostatní komponenty, je potřeba výslednou grafickou podobu okna z obrázku 10 mírně doladit.


Obrázek 10: JavaFX - Dialogové okno s Hello World

Stejně tak jako u JavaFX, neexistuje mezi Silverlight komponentami dialogové okno, které by šlo vytvořit jedním příkazem. Programátor si tak musí vytvořit celou komponentu sám.

```
<UserControl x:Class="PrezentaceSliverlight.OknoHelloWorld"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Width="200"
    Height="150">
    <Border BorderBrush="Black" BorderThickness="4" Width="200"
        Height="150">
        <Canvas Background="Whitesmoke">
            <Rectangle
                Width="193" Height="25" Stroke="Black"
                StrokeThickness="1" Fill="White" />
            <TextBlock
                Canvas.Top="5" Canvas.Left="5" Text="Okno s Hello World" />
            <TextBlock
                Canvas.Top="30" Canvas.Left="65" Text="Hello World!!" />
        </Canvas>
    </Border>
</UserControl>
```

Výpis 4: Silverlight - Dialogové okno s Hello World

Zdrojový kód může mít následující podobu (výpis 4). Okno se skládá ze základní komponenty *Canvas*, ke které je přidán okraj. Obsah okna je vytvořen jedním obdélníkovým rámováním a dvěma textovými poli. Jednoduché okno, které však bez dalších úprav je podstatně méně graficky zpracované než například u okna Flexu nebo GWT, je zobrazeno na obrázku 11.


Obrázek 11: Silverlight - Dialogové okno s Hello World

OpenLaszlo knihovna již komponentu dialogového okna obsahuje. Programátor tedy pouze doplní příslušné atributy a o zbytek se postará již předem připravený kód.

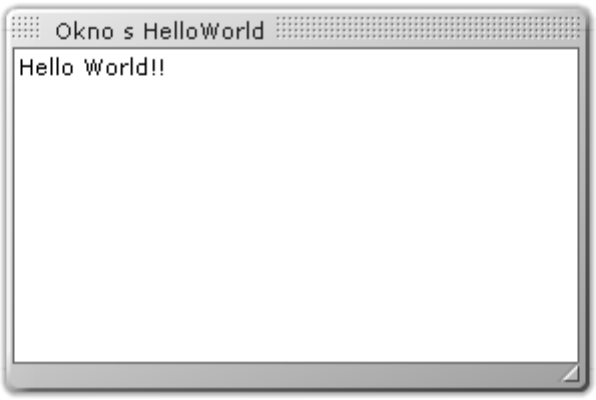
```
<class name="oknoHelloWorld" extends="window" x="100" y="100" width="300"
    height="200" title="Okno s HelloWorld" resizable="true">
    <text>Hello World!!</text>

    <animator attribute="width" from="100" to="300"
        duration="500" motion="linear"/>
    <animator attribute="height" from="100" to="200"
        duration="500" motion="linear"/>

    <simplelayout />
</class>
```

Výpis 5: OpenLaszlo - Dialogové okno s Hello World

Kromě základních parametrů jako jsou výška a šířka okna spolu s obsahem v podobě textu *Hello World!!* je do okna přidána dvojce prvků, které se starají o animaci okna. Tedy o jeho zobrazení jako rozpínající se okno. Tato část je nepovinná a ukazuje jen možnost, jak lze řešit funkcionalitu dialogového okna, kterou například okno GWT má zakomponovánu v základu. Stylově jednoduchý, kompaktní design je zobrazen na obrázku 12.




Obrázek 12: OpenLaszlo - Dialogové okno s Hello World

### 5.2.3 Klient-server komunikace

Stejně jako pro implementaci Hello World části jednotlivých RIA aplikací, byla pro komunikaci klient-server zvolena základní metoda posílání a získávání dat. Testování je provedeno na formuláři, který by mohl sloužit pro autentizaci uživatele. Cíl testu je jednoduchý: ověřit, zdali přihlašovaný uživatel nemá příjmení novak. Pokud ano, server odpoví kladně.

Pro ilustraci jsou uvedeny pouze zdrojové kódy klientské a serverové části. Výsledné prezentace jsou uvedeny v příloze na obrázcích 22, 23, 24, 25 a 26.


Jako základní komunikační technologie Flexu byla zvolena technologie Webových služeb.

```
<mx:Label x="238" y="106" text="Příjmení" width="198"/>
<mx:TextInput id="prijmeni" x="238" y="132"/>

<mx:Button y="175" label="Ověř totožnost!" enabled="true" width="200"
    height="30" x="238" click="overTotoznost()"/>
<mx:Label id="popisekOvereni" x="238" y="213"
    text="(příjmení musí být novak)" width="198"/>

<mx:WebService id="service"  result="onResult(event)" fault="onFault(event)"
   wsdl="http://localhost:8080/PrezentaceSilverlightService/wsdl/Service.wsdl"/>
```

Výpis 6: Flex - Klientská strana komunikace klient-server, UI

První část zdrojového kódu je vytvoření komponent a odkazu na Webovou službu. Komponenty tvoří jednoduchý formulář o jednom poli pro zadání příjmení, tlačítku pro odeslání dat na server a popisku, do kterého se zobrazí výsledek ověření. Dále je vytvořen odkaz na Webovou službu, který ukazuje na konkrétní deskriptor a také popisuje, jaké akce se zavolají na úspěšnou či neúspěšnou komunikaci.

Výpis 7 znázorňuje část ActionScriptu pro zpracování výsledku komunikace. Po zavolání metody *overTotoznost()*, kterou vyvolá stisk tlačítka pro ověření totožnosti, je zavolána Webová služba s požadavkem na operaci ověření totožnosti. Jako parametr je předána hodnota textového pole *prijmeni*. Pokud se komunikace a operace zdaří, je zavolána metoda *onResult()*, která nastaví popisku výsledek z volání, což může být v případě ukázkové Webové služby na ověření totožnosti buď text *ověřeno* nebo *neověřeno*. Při chybě komunikace nebo jiné chybě je zavolána metoda *onFault()*, která zobrazí dialogové okno s popisem chyby.

Webová služba, kterou volá klient ukázkové Flex aplikace je totožná s Webovou službou pro Silverlight ukázkovou aplikaci. Tímto využitím jedné implementace je znázorněn příklad využití jedné serverové strany různými klientskými stranami.

```
<mx:Script>
    <![CDATA[
        import mx.rpc.events.FaultEvent;
        import mx.rpc.events.ResultEvent;
        import mx.controls.Alert;

        private function onResult(event: ResultEvent): void {
            popisekOvereni.text=event.result.toString();
        }

        private function onFault(event: FaultEvent): void {
            Alert.show(event.fault.message.toString(), "CHYBA");
        }

        private function overTotoznost(): void {
            service.overTotoznost.send(prijmeni.text);
        }
        ...
    ]]>
</mx:Script>
```

Výpis 7: Flex - Klientská strana komunikace klient-server, logika

Pomocí RPC posílá data z klienta na server a zpět GWT. Implementace jednoduchého ověření dat z formuláře se skládá ze dvou rozhraní, implementace procedury na serveru a několik řádků kódu na klientovi. Zdrojové kódy jsou ve výpisech 8 až 11

```
package cz.kovar.client.rpc;

import com.google.gwt.user.client.rpc.RemoteService;

public interface RPCServiceGWT extends RemoteService {

    public String overTotoznost(String prijmeni);

}
```

Výpis 8: GWT - Rozhraní pro volání vzdálených procedur na straně klienta

Základem komunikace je synchronní rozhraní (výpis 8). Toto rozhraní je potomkem *RemoteService*, což znamená, že jej lze použít k RPC komunikaci a musí jej implementovat serverová strana.

```
package cz.kovar.client.rpc;

import com.google.gwt.user.client.rpc.AsyncCallback;

public interface RPCServiceGWTAsync {
    void overTotoznost(String prijmeni, AsyncCallback<String> callback);
}
```

Výpis 9: GWT - Rozhraní pro volání vzdálených procedur na straně klienta, asynchronní část

Jelikož celá komunikace je asynchronní, musí existovat asynchronní část, tedy rozhraní, které obsahuje metody založené na synchronní verzi. Metody tohoto rozhraní nevrací žádnou hodnotu, avšak jako povinný parametr obsahují rozhraní *AsyncCallback*<*T*>, které zajišťuje zpracování odpovědi a tedy i vrací případná data, podobně jako je tomu u metod synchronního rozhraní. Generický parametr *T* udává, jakého typu budou vrácená data.

Ve výpisu 10 je uveden kód pro zavolání vzdálené procedury z klienta. Nejdříve je vytvořen proxy objekt synchronního rozhraní. Jelikož je však komunikace asynchronní, je výsledný proxy objekt přetypován jako asynchronní. Přetypování je bezpečné, jelikož vytvořený proxy objekt automaticky implementuje asynchronní rozhraní. Z tohoto důvodu je však nutné dodržovat jmenné konvence, tedy synchronní rozhraní musí mít odpovídající asynchronní část s *Async* koncovkou.

Metoda *overTotoznost(String data)*, která je volána po stisku tlačítka, přepošle volání na vytvořený proxy objekt. Zde je nutno implementovat již zmíněné rozhraní, které poskytuje dvě metody. Pro zpracování pozitivního výsledku, nebo negativního. Implementace těchto metod je nasnadě. Při chybě je zobrazeno okno s chybovým hlášením, pokud se RPC volání podaří, je popisku pro zobrazení ověření nastavena hodnota parametru metody *onSuccess(String zprava)*, která představuje data vrácená z předchozího požadavku.

Kód z výpisu 11 znázorňuje implementaci serverové RPC komunikace. Serverová část je v podstatě servlet, který pouze implementuje synchronní rozhraní pro RPC komunikaci. Dědí proto z *RemoteServiceServlet* a implementuje konkrétní rozhraní. Implementace je přímočará: pokud se uživatel přihlašuje jako *novak*, server odpoví kladně, v opačném případě vrátí negativní odpověď.

```
public class PrezentaceGWT implements EntryPoint {

    private static final MyConstants MESSAGES = (MyConstants) GWT
            .create(MyConstants.class);

    final Label priklad2LabelStatus = new Label(MESSAGES.priklad2LabelStatus());

    private RPCServiceGWTAsync serviceGWTAsync = (RPCServiceGWTAsync) GWT
            .create(RPCServiceGWT.class);

    public void onModuleLoad() {
        // ... inicializace
    }

    void overTotoznost(String data) {
        serviceGWTAsync.overTotoznost(data, new AsyncCallback<String>() {
            public void onFailure(Throwable e) {
                Window.alert(MESSAGES.chybaRPC() + e.getMessage());
            }
            public void onSuccess(String zprava) {
                priklad2LabelStatus.setText(zprava);
            }
        });
    }
```

Výpis 10: GWT - Volání vzdálené procedury na straně klienta

```
package cz.kovar.server;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

public class RPCServiceGWTImpl extends RemoteServiceServlet implements
        RPCServiceGWT {

    private static final long serialVersionUID = -4637263140056720082L;

    public String overTotoznost(String prijmeni) {
        if ("novak".equals(prijmeni)) {
            return "ověřeno";
        }
        return "neověřeno";
    }

}
```

Výpis 11: GWT - Implementace serverové části RPC

Získávání dat z Webové služby bylo zvoleno jako ukázka i u JavaFX technologie. Díky použitému NetBeans IDE lze vytvořit jak Webovou službu, tak klientskou část téměř automaticky. Programátor píše pouze klientskou část kódu. Serverovou část včetně rozhraní samotné Webové služby lze vygenerovat pomocí průvodců. Kód implementace operace Webové služby pro ověření totožnosti je zobrazen ve výpisu 12.

```
@WebService()
public class WebServiceExample {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "overTotoznost")
    public String overTotoznost(@WebParam(name = "prijmeni") String prijmeni) {
        if ("novak".equals(prijmeni)) {
            return "ověřeno";
        }
        return "neověřeno";
    }
}
```

Výpis 12: JavaFX - Implementace operace ověření totožnosti Webové služby

Klientská část aplikace, je rozdělena na dvě části. Aby mohl JavaFX klient volat Webovou službu, potřebuje k tomu nějakého prostředníka, klienta, který službu zavolá a vrátí výsledek. Takového klienta lze implementovat jako sadu tříd, které se starají o spojení s Webovou službou a zpracování výsledku volání. I tuto část lze vytvořit v NetBeans IDE automaticky. Výpis 13 znázorňuje již pouze třídu, která poskytuje přístupový bod připojené Webové služby.

Kód JavaFX Skriptu, který implementuje volání Webové služby je ve výpisu 14. Pomocný klient pro přístup k Webové službě je do JavaFX aplikace přidán jako externí knihovna. Lze jej však jednoduše zavolat a vytvořit tak přístupový bod pro zavolání konkrétní operace. Jako parametr metody pro ověření totožnosti je předán text z formulářového pole. Chyba při komunikaci je zachycena jako výjimka, která může být dále zpracována.

```
public class ConnectionHelper {

    public static ws.WebServiceExample getWSConnection() {
        try {
            ws.WebServiceExampleService service =
                    new ws.WebServiceExampleService();
            return service.getWebServiceExamplePort();

        } catch (Exception ex) {
            Logger.getLogger(ConnectionHelper.class.getName()).
                    log(Level.SEVERE, null, ex);
        }
        return null;
    }
}
```

Výpis 13: JavaFX - Pomocná třída pro přístup k Webové službě

```
function () {
    try {
        var remoteServer: WebServiceExample =
        ConnectionHelper.getWSConnection();

        var results = remoteServer.overTotoznost(textFieldPrijmeni.text);
        if (results != null) {
            odpoved = results;
        } else {
            odpoved = ##"zadnaOdpoved"
        }
    } catch (e:Exception) {
        System.out.println("exception: {e}");
    }
}
```

Výpis 14: JavaFX - Implementace volání Webové služby na klientovi

Komunikace klient-server v Silverlightu je rovněž postavená nad voláním Webové služby. Obdobně jako u předchozích implementací, je Webová služba implementována jako Java třída. WSDL (Web Services Description Language) deskriptor vygeneruje Eclipse

IDE automaticky s vytvořením služby. Klienta lze propojit s vygenerovanou Webovou službou také automaticky. IDE vytvoří pomocné propojovací objekty a při psaní klientského kódu tak programátor volá již vygenerované metody.

```
public class Service {

    public String overTotoznost(String param) {
        if ("novak".equals(param)) {
            return "ověřeno";
        }
        return "neověřeno";
    }
}
```

Výpis 15: Silverlight - POJO jako představitel Webové služby

Implementace jednoduché Java třídy, která po obalení do Webové služby slouží pro ověření totožnosti je zobrazena ve výpisu 15. Deskriptor WSDL je pak uveden jako obrázek 28 v příloze.

```
private void overTotoznost(object sender, RoutedEventArgs arg1)
{
    var client = new DataService.ServiceClient();
    client.overTotoznostCompleted +=
      new EventHandler<DataService.overTotoznostCompletedEventArgs>(client_overTotoznost);
    client.overTotoznostAsync(textBoxPrijmeni.Text);
}

private void client_overTotoznost(object sender,
            DataService.overTotoznostCompletedEventArgs e)
{
    overTotoznostLabel.Text = e.Result;
}
```

Výpis 16: Silverlight - Implementace volání Webové služby na klientovi

Implementace volání Webové služby je znázorněna ve výpisu 16. Vše je implementováno v kódu na pozadí *Page.xaml.cs*, který slouží pro zpracování logiky klienta. Implementace se skládá ze tří částí. Nejdříve je vytvořen proxy objekt, kterému je následně přidán posluchač asynchronního volání. Jako parametr je mu předán odkaz na metodu, která bude vykonána v případě kladné odezvy. Třetí částí je implementace samotného asynchronního volání. Jako parametr je tentokrát předán konkrétní řetězec k ověření. Metoda pro zpracování odpovědi pak již jen nastaví vrácenou hodnotu příslušné komponentě, v tomto případě popisku pro zobrazení ověření. Posledním výpisem 17 je jen UI komponenta tlačítka, která volá metodu pro vytvoření a zpracování požadavku na Webovou službu.

```
<Button Content="Ověř totožnost!" Width="200" Height="30"
    Click="overTotoznost" />
```

Výpis 17: Silverlight - Kód tlačítka, po jehož stisku je volána Webová služba ověření totožnosti

Klient-server komunikace v ilustračním příkladu technologie OpenLaszlo je řešena pomocí RPC. Konkrétně se jedná o JavaRPC, kde je serverová část implementována jako POJO, obdobně jako je tomu u Silverlightu. Zdrojový kód je tedy naprosto shodný jako ve výpisu 15.

Ve výpisu 18 je uvedena část kódu pro vytvoření proxy objektu na klientovi. Jeden z povinných atributů je právě název POJO třídy a dále jsou definovány obsluhy různých typů událostí. Důležitá je obsluha události *ondata* pro zpracování výsledku RPC volání pokud nedojde k žádné chybě a *onerror*, která zpracovává právě chyby při RPC komunikaci. Tyto obsluhy jsou použity v případě, že nejsou definovány u objektu vzdáleného volání, jehož kód je ve výpisu 19. Povinnými parametry jsou název, pod kterým bude volání dostupné a kontext, což je odkaz na proxy RPC objekt. Poslední část, tedy tag *param* slouží k předání parametrů RPC volání, což je v tomto případě text z formulářového pole příjmení.

Kód tlačítka pro vyvolání RPC je uveden ve výpisu 20. Tlačítko pouze na událost kliknutí spustí RPC volání.

```
<javarpc name="service" scope="session" autoload="true"
        remoteclassname="rpc.RPCServiceOpenLaszlo">

    <handler name="onload">
        Debug.debug("RPC service loaded.");
        Debug.debug("%w", this.proxy);
    </handler>

    <handler name="ondata" args="res">
        canvas.formular.detail.status.setAttribute('text', res);
    </handler>

    <handler name="onerror" args="err">
    canvas.formular.detail.status.setAttribute('text', "CHYBA RPC KOMUNIKACE");
    </handler>

</javarpc>
```

Výpis 18: OpenLaszlo - Kód vytvoření proxy objektu pro RPC komunikaci

```
<remotecall funcname="overTotoznost" remotecontext="$once{canvas.service}" >
    <param value="${canvas.formular.detail.prijmeni.value}" />
</remotecall>
```

Výpis 19: OpenLaszlo - Kód RPC volání

```
<button text="Ověř totožnost!" height="30" width="200">
    <handler name="onclick">
        canvas.overTotoznost.invoke();
    </handler>
</button>
```

Výpis 20: OpenLaszlo - Kód tlačítka, které spouští RPC volání

### 5.2.4 Multimédia

Pro demonstraci podpory multimédií byla vybrána krátká videa, která jsou přehrána v jednoduchých přehrávačích. Implementováno je základní načtení a přehrání videosekvence spolu s ovládacími prvky, tedy tlačítky přehrát, pozastavit a ukončit přehrávání.

Videa bylo nutno převést do konkrétních formátů pro jednotlivé technologie. U většiny postačil formát FLV, jedině u Silverlightu bylo potřeba vytvořit videosekvenci ve WMV formátu. V následujících výpisech jsou uvedeny zdrojové kódy pro načtení datového zdroje a vytvoření přehrávače spolu s kódy ovládacích prvků.

Ve výpisech není technologie GWT, ke které není přehrávání videa implementováno.


```
<mx:VideoDisplay x="520" y="67" width="320" height="240"
                         autoPlay="false" id="video">
    <mx:source>ovce.flv</mx:source>
</mx:VideoDisplay>

<mx:Button x="520" y="324" label="Play" click="video.play()"/>
<mx:Button x="593" y="324" label="Pause" click="video.pause()"/>
<mx:Button x="683" y="324" label="Stop" click="video.stop()"/>
```

Výpis 21: Flex - Kód přehrávače videa a ovládacích tlačítek

Kód přehrávače ve Flexu (výpis 21) je velice jednoduchý. Komponenta je obsažena v knihovnách a stačí ji tedy pouze vložit do klientského kódu a nastavit příslušné parametry. Hlavním, kromě pozice a velikosti přehrávacícho okna, je odkaz na samotné video. Parametr *autoPlay* pouze určuje, zda bude video automaticky spuštěno při načtení. Zde je vhodné poznamenat, že první snímek videa je zobrazen již při načtení a nezobrazí se tak pouze prázdná obrazovka, jak je tomu u přehrávačů jiných technologií.

Kód ovládacích prvků je prostý. Jsou to pouze tlačítka, která na kliknutí reagují příslušnou akcí, tedy přehráním, pozastavením, či úplným zastavením přehrávání.

```
var hostPortString = ##"hostPort";
var mediaPlayer : MediaPlayer = MediaPlayer {
    autoPlay: false;
    media: Media {
        source: "{hostPortString}/videa/housenka.flv";
    }
}

var mediaView = MediaView {
    preserveRatio: true
    visible: true;
    mediaPlayer: mediaPlayer
}
```

Výpis 22: JavaFX - Kód přehrávače videa

```
var tlacitkoPause = SwingButton {
    width: 150;
    text: bind textTlacitkaPause;
    action: function() {
        if (mediaPlayer.paused == false) {
            textTlacitkaPause = "Unpause";
            mediaPlayer.pause()
        } else {
            textTlacitkaPause = "Pause";
            mediaPlayer.play()
        }
    }
}
```

Výpis 23: JavaFX - Kód tlačítka pro pozastavení přehrávání

Samotný přehrávač videa, jehož kód je uveden ve výpisu 22, se skládá ze tří částí. První a základní je element *Media*, který definuje odkaz na samotné video. Druhým prvkem je samotný přehrávač *MediaPlayer*, který se stará o samotné zpracování a přehrávání datového zdroje. Třetí komponentou je *MediaView*, která se stará o zobrazení samotného přehrávače a poskytuje přídavné funkce přehrávání.

V ukázce zdrojového kódu (výpis 23) je uvedeno tlačítko pro pozastavení přehrávání. Rozšířením, oproti stejnému tlačítku Flexu, je pouze změna textu při různých stavech přehrávání.

```
<MediaElement x:Name="media" Width="300" Height="300"
    AutoPlay="false" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3" />

<Button Click="StopMedia" Width="50" Grid.Column="0"
    Grid.Row="2" Content="Stop" />
<Button Click="PauseMedia" Width="50" Grid.Column="1"
    Grid.Row="2" Content="Pause" />
<Button Click="PlayMedia" Width="50" Grid.Column="2"
    Grid.Row="2" Content="Play" />
```

Výpis 24: Silverlight - Kód UI přehrávače a tlačítek

Ve výpisu 24 je uveden kód UI pro přehrávač v Silverlightu. Stejně jako je u Flexu, existuje jedna komponenta *MediaElement*, která se stará jak o zpracování videa tak jeho zobrazení. Kód je doplněn o ukázku implementace klasických tří ovládacích tlačítek.

V kódu na pozadí je poté uveden odkaz na datový zdroj pro přehrávač a metody pro obsluhu událostí kliknutí na jednotlivá ovládací tlačítka. Kód je uveden ve výpisu 25. Metody pro pozastavení a zastavení přehrávání jsou stejné, liší se pouze v názvu a volání příslušné metody na komponentu přehrávače. Jsou tedy pro jednoduchost ve výpisu vynechány.

```
public Page()
{
    InitializeComponent();
    media.Source = new System.Uri(Application.Current.Host.Source, "../videa/zelvicka.wmv");
}

private void PlayMedia(object sender, RoutedEventArgs e)
{
    media.Play();
}
```

Výpis 25: Silverlight - Kód na pozadí pro ovládání videa

Kód pro implementaci přehrávače videa se v OpenLaszlu skládá ze dvou částí. Jak je vidět z výpisu 26, první částí je komponenta *mediastream*, která se tará o načtení a zpracování samotného videa. Druhá komponenta *videoview* pak jen zapouzdřuje datovou komponentu a představuje samotný přehrávač.

```
<mediastream name="mediastream" autoplay="true" type="http">
    <handler name="oninit">
        this.setAttribute("url", pathToVid + "/videa/vcelky.flv");
    </handler>
</mediastream>

<videoview name="prehravac" type="http" autoplay="true" width="320"
    height="240" stream="canvas.video.mediastream"/>
```

Výpis 26: OpenLaszlo - Kód přehrávače videa

V ukázkovém příkladu je nastaveno automatické přehrávání videa po načtení. Pokud je tato volba vypnuta, nezobrazí se první snímek z přehrávaného videa, ale pouze prázdné okno. Z tohoto důvodu jsou v příkladu použity pouze dvě tlačítka, která podle stavu přehrávání poskytují možnost video pozastavit či přehrát znovu. Zdrojové kódy těchto tlačítek jsou uvedeny ve výpisu 27.

```
<button text="Replay">
    <handler name="onclick">
        if (!canvas.video.prehravac.stream.paused) {
            canvas.video.prehravac.stream.play()
        }
    </handler>
</button>

<button name="pauseButton" text="Pause">
    <handler name="onclick">
    if (!canvas.video.prehravac.stream.paused) {
        canvas.video.prehravac.stream.setAttribute('paused', true);
        canvas.video.tlacitka.pauseButton.setAttribute('text', "Continue");
    } else {
        canvas.video.prehravac.stream.setAttribute('paused', false);
        canvas.video.tlacitka.pauseButton.setAttribute('text', "Pause");
    }
    </handler>
</button>
```

Výpis 27: OpenLaszlo - Kód ovládacích tlačítek

## 5.3 Typická aplikace v prostředí Internetu

Za typickou aplikaci v prostředí Internetu lze považovat YouTube. Aplikace, která stojí na pomezí webových a RIA aplikací. Má bohatý obsah, který je však načítán synchronně a s každým požadavkem na server je generován nový obsah. Pouze některé z menších funkcí aplikace jsou napsány asynchronně. Její hlavní cíl je zobrazit a přehrát multimediální obsah. Bohaté UI je však tvořeno z větší části HTML a logiku zajišťuje JavaScript.

V následující části bude nastíněna implementace takovéto aplikace pomocí RIA technologie.

### 5.3.1 Požadavky na aplikaci

Ze tří nejvhodnějších kandidátů, tedy Flexu, Silverlightu a OpenLaszla, kteří "vyhráli" srovnávací část, lze vybrat kteroukoliv technologii. Jelikož jsou RIA technologie zaměřeny především na UI, zajímavým řešením je zcela určitě FlexYouTube, tedy YouTube napsaný ve Flexu.

Aplikace je poměrně rozsáhlá a mezi její základní funkce lze zařadit následující:

- Registrace uživatelů - do aplikace se lze přihlásit pod svým jménem/heslem a následně spravovat svá videa, vytvářet playlisty, či komentovat a hodnotit videa ostatních uživatelů.
- Přehrávání videa - jak pro registrované, tak pro neregistrované uživatele umožňuje umožňuje přehrát videa z široké nabídky tematických okruhů.
- Kolekce a organizace videa - YouTube aplikace nejen shromažďuje uživateli nahraná videa, ale také je třídí a zařazuje do kategorií.

Aplikace má dále spoustu drobnějších funkcí, jako je již zmíněná možnost hodnotit videa, dále sdílet videa mezi uživateli, vytvářet žebříčky nejpopulárnějších videí a další. Přeskočíme-li registraci uživatele, bez jehož videa by nebylo co přehrávat, a její implementace, která spočívá pouze v připravení formuláře pro odeslání několika málo dat do DB, lze z této bohaté funkcionality vybrat právě přehrávání videa jako odrazový bod. Jednotlivé části implementace jsou nastíněny v následujících dvou podkapitolách.

### 5.3.2 Klient

Jako pomůcku při ilustraci implementace použijme obrázek 27 z přílohy. Na tomto obrázku je zobrazena většina UI aplikace, potřebná k přehrání videa. Pomineme-li jednoduchost a pouze základní nástin implementace, tak je z obrázku patrné, že lze pomocí komponent Flexu vytvořit docela věrnou kopii UI originálu. Některé prvky, jako například rozbalovací seznam s dalšími video soubory jsou odlišné, avšak plní v základu stejnou funkci jako v předloze.

Jelikož lze celé UI implementovat jednoduše ve Flex Builder IDE, vyjmenujme některé zásadní problémy a řešení, které s implementací rozsáhlého UI souvisí:

- Stránkování - jeden z důležitých aspektů vývoje aplikace. Většina webových aplikací řeší stránkování přechodem na novou stránku, či přegenerování obsahu novými daty. Ve Flashi nic takového možné není a je potřeba generovat obsah dynamicky podle požadovaných změn. Pro implementaci změny obsahu určité oblasti či celé aplikace lze využít některou z předpřipravených komponent. Například komponentu *Accordion*, *ViewStack* nebo *TabNavigator*.
- Login - přihlášení do aplikace se většinou děje na samostatné stránce a formuláři, kde po úspěšné autentizaci dojde k přesměrování na hlavní stránku aplikace. Toto lze řešit buď výše uvedenou dynamickou změnou obsahu, nebo statickou stránkou s částí klienta, tedy přihlašovacím formulářem a následným přesměrováním na stránku s hlavní aplikací.

### 5.3.3 Klient-server komunikace

Vhodným prostředkem pro komunikaci se serverem mohou být jak Webové služby, tak RPC. Pro jednoduché funkce, jako je již zmíněná autentizace uživatele, či hlasování nebo hodnocení shlédnutých videí, stačí jednoduchá implementace asynchronního volání serveru. Uživatel může v sledovat video, zatím co je zpracován jeho vložený komentář a "přišpendlen" zpět do seznamu. Stejně tak lze jednoduše nahrát ze serveru seznam uživatelových videí, které se zobrazí po načtení v příslušné komponentě. To vše se děje na pozadí.

Zajímavější je však samotné nahrání, takzvaný upload videa na server. Zde jen uvedeme, že pro ukládání videa na serveru nemusí sloužit databáze, ale postačí souborový systém s příslušnou adresářovou strukturou. Nahrané soubory lze dobře třídit například podle času a i ID uživatele, jelikož je téměř nemožné nahrát dva soubory od jednoho uživatele v jeden čas. Ke každému souboru lze pak vygenerovat jednoznačný deskriptor, podle kterého jej server rozpozná a příslušné video vrátí klientovi.

Výše uvedené řádky v podstatě popisují uložení a zpracování videa na serveru. Z druhé strany, tedy od klienta to vypadá následovně:

- Upload - uživatele zadá v UI cestu k videu a potvrdí odeslání.
- Odeslání na server - data jsou odeslána na server a UI ohlásí uživateli zprávu o probíhajícím zpracování nahraného videa. Uživatel je pouze informován, avšak nic mu nebrání dále pracovat s aplikací.
- *Zpracování videa na serveru* - tato operace představuje: vytvoření příslušného adresáře, konverzi videa do formátu FLV a vytvoření deskriptoru nahraného zpracovaného souboru.
- Konec uploadu - UI zobrazí klientovi hlášení o průběhu a výsledku zpracování videa serverem a aktualizuje příslušné seznamy (playlisty).

Tento zjednodušený model obsahuje základní akce, které jsou provedeny při uploadu videa. Implementace serverové strany, at' už je to PHP, ASP.NET nebo Java, zajistí příslušnou konverzi a uložení nahraného videa a UI pouze zobrazí výsledek.

# 6 Závěr

Cílem této práce bylo prostudování a srovnání RIA technologií pro vývoj aplikací v Internetu. Výsledkem je komplexní porovnání jednotlivých vlastností RIA technologií a to z pohledu obecných vlastností, funkcionality a z pohledu uživatele. Jednoduché příklady pak doplňují srovnávací část práce a ilustrují základní prvky RIA technologií.

Práce se skládá ze dvou hlavních částí. Při studování jednotlivých RIA technologií jsem použil znalosti z vývoje webových aplikací a znalosti jednotlivých programovacích jazyků. Srovnávací část je vyústěním základních otázek ohledně vývoje aplikace v Internetu. Ilustrační příklady jsou vytvořeny tak, aby je bylo možno spustit na jakémkoliv počítači s podporou běhu webových aplikací, tedy s jednoduchým webovým serverem a prohlížečem.

Námětem dalších prací může být například srovnání RIA technologií s dnes běžně používanými vývojovými rámci jako je například Apache Struts, Spring MVC, Ruby on Rails nebo Tapestry. Dalším námětem může být také rozšíření základní kostry aplikace FlexYouTube o další funkcionalitu.

Jelikož se RIA technologie v čase neustále mění a vznikají nové verze, může být otázka budoucnosti vývoje aplikací v Internetu, tedy kterým směrem se vývoj vydá a která technologie povede při vývoji RIA aplikací, taktéž zajímavým tématem další diplomové práce.

# 7 Literatura


[1] AHMED, Tariq, HIRSCHI, Jon, ABID, Faisal. *Flex 3 in Action*. Manning Publications Co., Leden, 2009. 576 s. ISBN 1-933988-74-6.

[2] ALLAIRE, Jeremy. *Macromedia Flash MX - A next-generation rich client* [online]. 2002. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf

[3] BELLARD, Fabrice. *FFmpeg* [online]. 2007. http://ffmpeg.mplayerhq.hu/index.html

[4] BERNARD, Bořek. *Rich Internet Applications v roce 2008*, [online]. 2008. http://interval.cz/clanky/rich-internet-applications-v-roce-2008/

[5] CAMPBELL, Chad A., STOCKTON, John. *Silverlight 2 in Action*. Manning Publications Co., Říjen, 2008. 400 s. ISBN 1-933988-42-8.

[6] ČERMÁK, Miloš. *Na velikosti záleží* [online]. 2008. http://www.lupa.cz/clanky/na-velikosti-zalezi.

[7] HANSON, Robert, TACY, Adam. *GWT in Action*. Manning Publications Co., Červen, 2007. 632 s. ISBN 1-933988-23-1.

[8] KLEIN, Norman, CARLSON, Max, MACEWEN, Glenn. *Laszlo in Action*. Manning Publications Co., Leden, 2008. 552 s. ISBN 1-932394-83-4.

[9] LOOSLEY, Chris. *Rich Internet Applications: Design, Measurement and Management Challenges* [online]. 2006. http://www.keynote.com/docs/whitepapers/RichInternet_5.pdf.

[10] Microsoft Corporation. *Kit3D* [online]. 2009. http://www.codeplex.com/Kit3D

[11] Microsoft Corporation. *Expression Blend 2* [online]. 2009. http://www.microsoft.com/expression/products/Overview.aspx?key=blend

[12] MORRIS, Simon. *JavaFX in Action*. Manning Publications Co., Srpen, 2008. 375 s. ISBN 1-933988-99-1.

[13] Novell, Inc. *Moonlight - mono* [online]. 2007. http://www.mono-project.com/Moonlight

[14] O'REILLY, Tim. *What Is Web 2.0* [online]. 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[15] Papervision3D. *Papervision3D*, [online]. 2009.
http://www.papervision3d.org/

[16] RADECKÝ, Michal, *Internetové Technologie* [online]. 2008.
http://www.cs.vsb.cz/radecky/files/itpres/it2008_2.pdf

[17] Wikimedia Foundation, Inc. *DARPA*, [online]. 2009.
http://en.wikipedia.org/wiki/Defense_Advanced_Research_Projects_Agency

[18] Wikimedia Foundation, Inc. *Enquire*, [online]. 2009.
http://en.wikipedia.org/wiki/Enquire

[19] Wikimedia Foundation, Inc. *History of the World Wide Web*, [online]. 2009.
http://en.wikipedia.org/wiki/History_of_the_World_Wide_Web

[20] Wikimedia Foundation, Inc. *Rich Internet application*, [online]. 2009.
http://en.wikipedia.org/wiki/Rich_Internet_application

[21] Wikimedia Foundation, Inc. *Tim Berners-Lee*, [online]. 2009.
http://en.wikipedia.org/wiki/Tim_Berners-Lee

[22] Wikimedia Foundation, Inc. *Vannevar Bush*, [online]. 2009.
http://en.wikipedia.org/wiki/Vannevar_Bush

[23] Wikimedia Foundation, Inc. *Web application*, [online]. 2009.
http://en.wikipedia.org/wiki/Web_application

[24] Wikimedia Foundation, Inc. *World Wide Web*, [online]. 2009.
http://cs.wikipedia.org/wiki/World_Wide_Web

# A Obsah přiloženého CD

**Bin**

| | |
|---|---|
| ***.war** | **- webová prezentace konkrétní RIA technologie** |

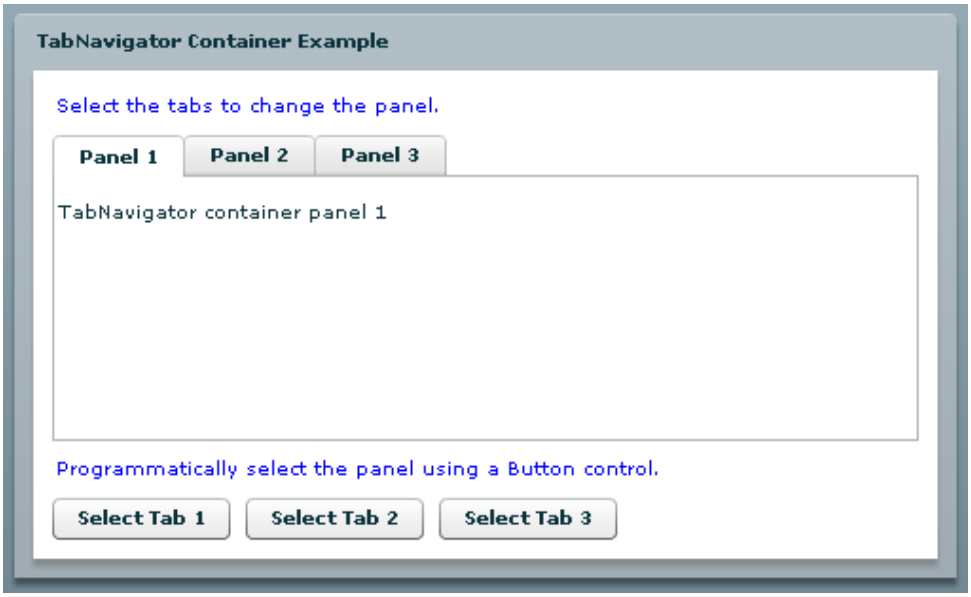
**Src**

| | |
|---|---|
| **Flex** | **- zdrojové soubory Flex prezentace a FlexYouTube aplikace jako projekty do Eclipse IDE** |
| **GWT** | **- zdrojové soubory GWT prezentace jako projekt do Eclipse IDE** |
| **JavaFX** | **- zdrojové soubory JavaFX prezentace a JavaFX Webové služby jako projekty do NetBeans IDE** |
| **OpenLaszlo** | **- zdrojové soubory OpenLaszlo prezentace a RPC služby jako projekty do Eclipse IDE** |
| **Silverlight** | **- zdrojové soubory Silverlight prezentace a Webové služby jako projekty do Eclipse IDE** |

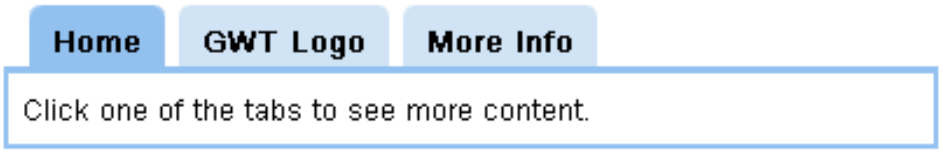
**Doc** **- zdrojové soubory textu diplomové práce**

# B Grafický přehled komponent srovnávaných technologií

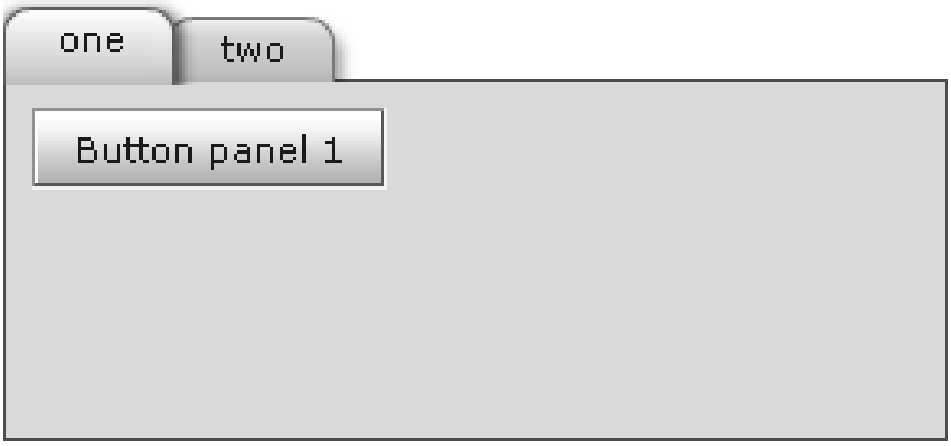
Panel záložek (TabNavigator, TabPanel, Tabs, TabControl)



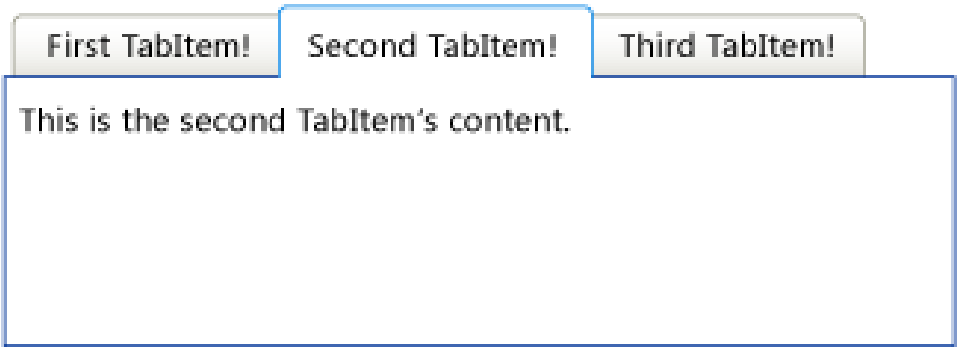

Obrázek 13: Flex - TabNavigator




Obrázek 14: GWT - TabPanel



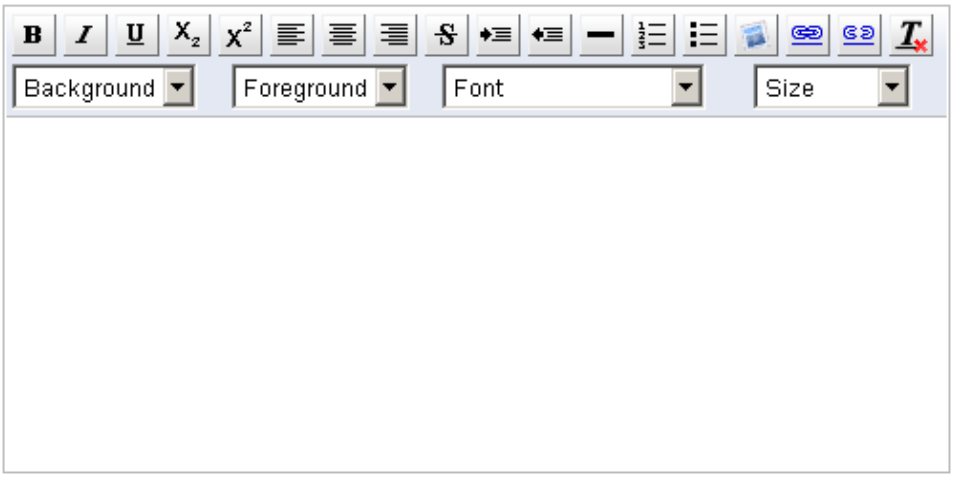

Obrázek 15: OpenLaszlo - Tabs




Obrázek 16: Silverlight - TabControl

Bohatý editor pro vkládání textu (RichTextEditor, RichTextArea)


Obrázek 17: Flex - RichTextEditor



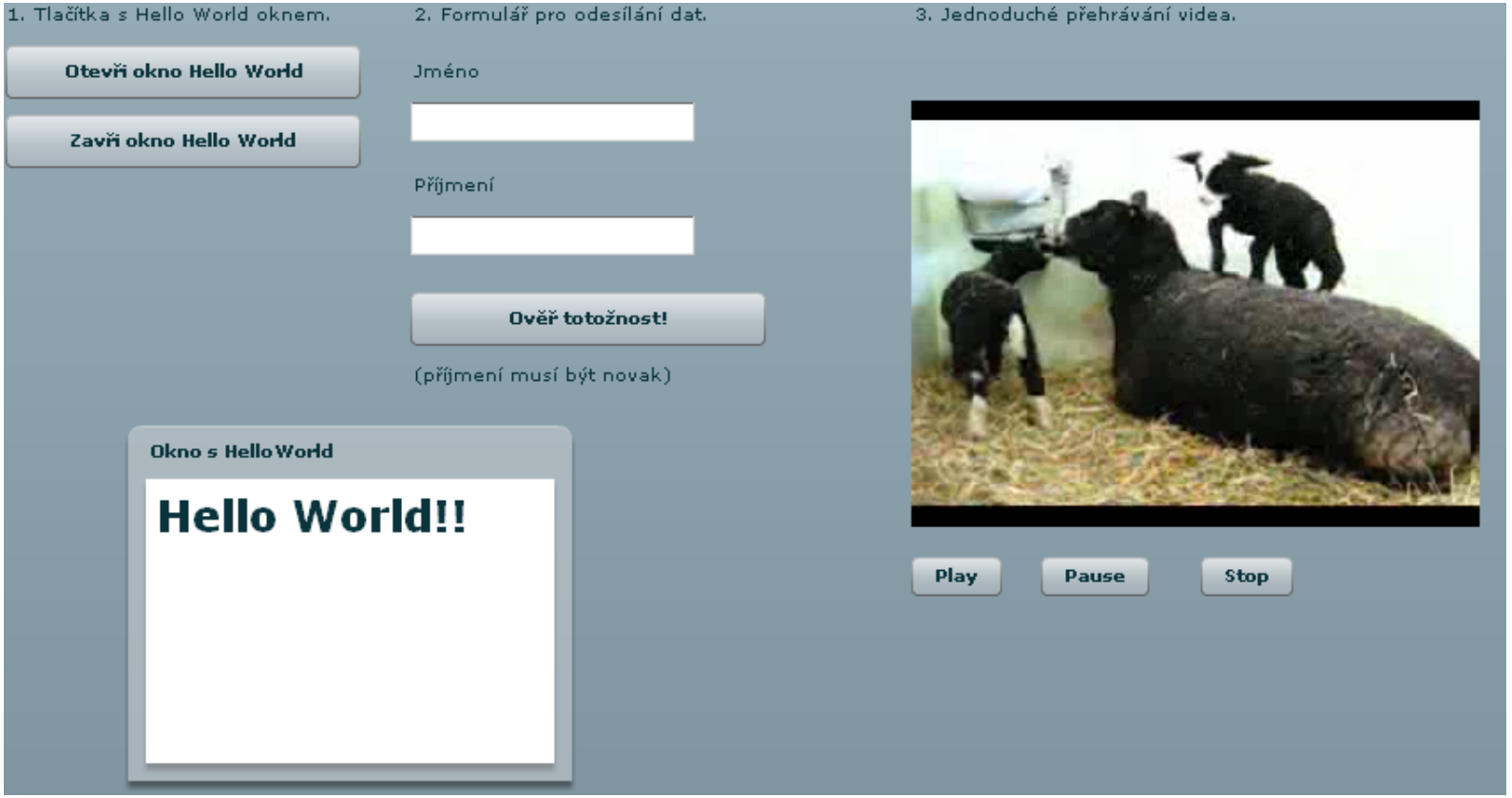

Obrázek 18: GWT - RichTextArea

Kalendář (DateField, CalendarControl, DatePicker)


Obrázek 19: Flex - DateField


Obrázek 20: Silverlight - CalendarControl

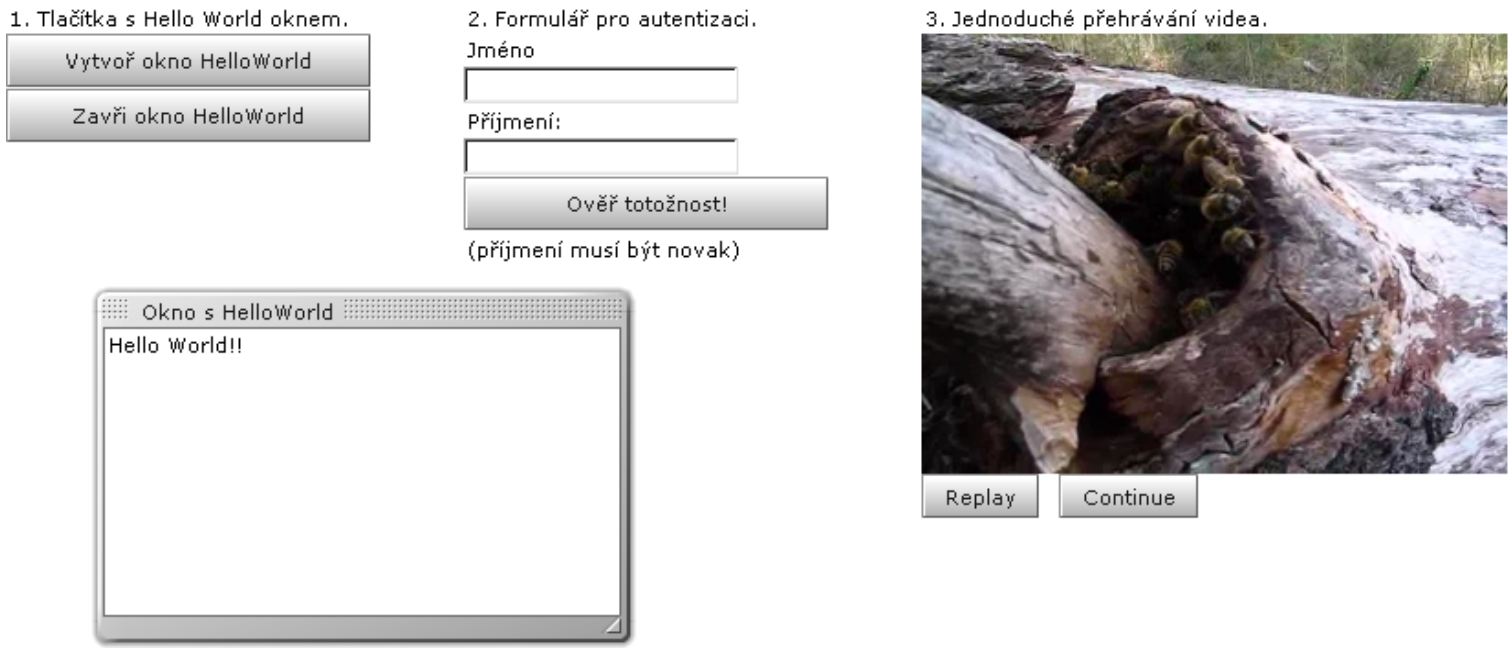

Obrázek 21: OpenLaszlo - DatePicker



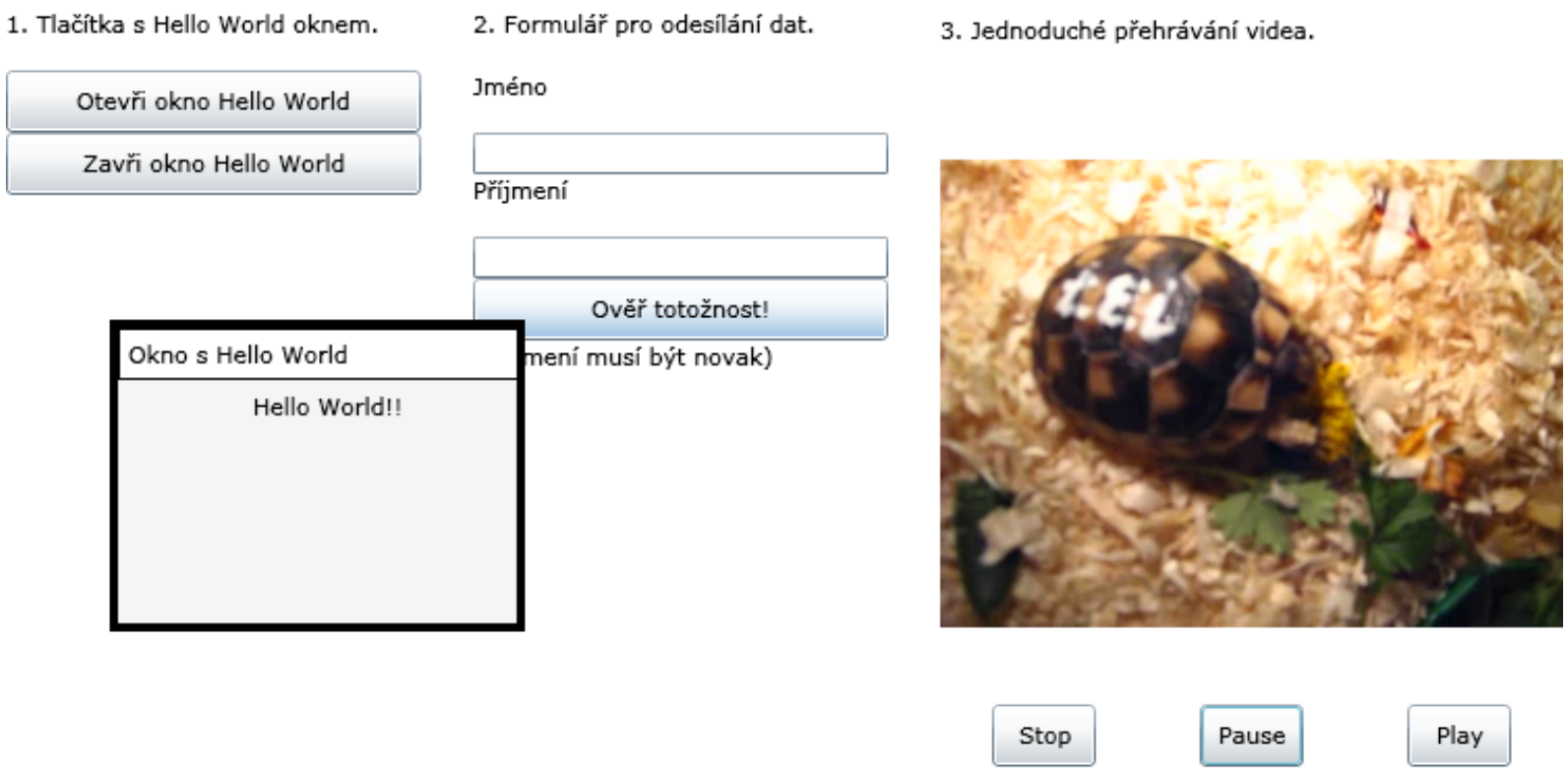

Obrázek 22: Prezentace UI technologie Flex


Obrázek 23: Prezentace UI technologie GWT

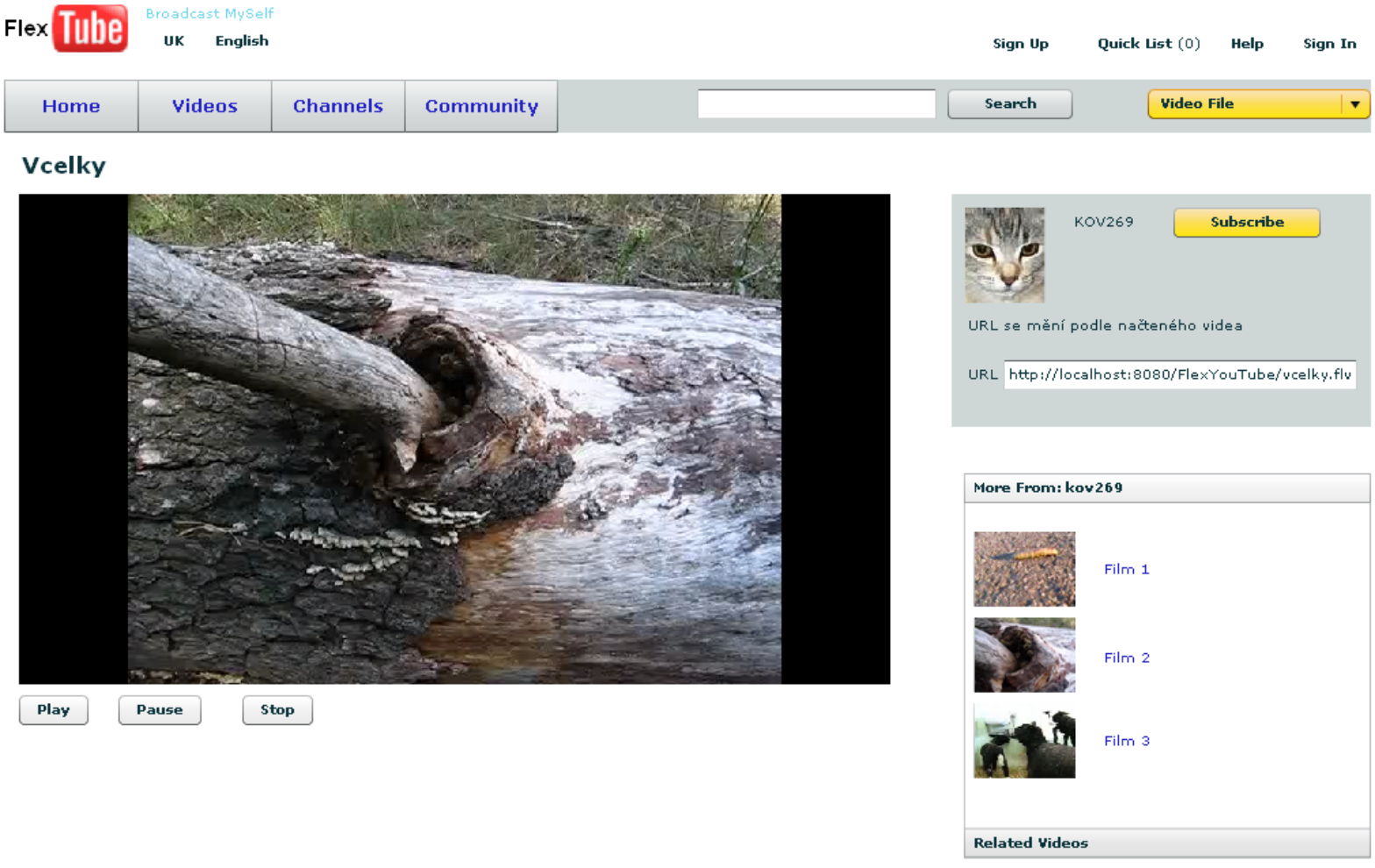

Obrázek 24: Prezentace UI technologie JavaFX




Obrázek 25: Prezentace UI technologie OpenLaszlo




Obrázek 26: Prezentace UI technologie Silverlight




Obrázek 27: Komplexní aplikace - FlexYouTube

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ws"
    xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://ws"
    xmlns:intf="http://ws" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!--
        WSDL created by Apache Axis version: 1.4 Built on Apr 22, 2006
        (06:55:48 PDT)
    -->
    <wsdl:types>
        <schema elementFormDefault="qualified" targetNamespace="http://ws"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="overTotoznost">
                <complexType>
                    <sequence>
                        <element name="param" type="xsd:string" />
                    </sequence>
                </complexType>
            </element>
            <element name="overTotoznostResponse">
                <complexType>
                    <sequence>
                        <element name="overTotoznostReturn" type="xsd:string" />
                    </sequence>
                </complexType>
            </element>
        </schema>
    </wsdl:types>
    <wsdl:message name="overTotoznostResponse">
        <wsdl:part element="impl:overTotoznostResponse" name="parameters" />
    </wsdl:message>
    <wsdl:message name="overTotoznostRequest">
        <wsdl:part element="impl:overTotoznost" name="parameters" />
    </wsdl:message>
    <wsdl:portType name="Service">
        <wsdl:operation name="overTotoznost">
            <wsdl:input message="impl:overTotoznostRequest"
             name="overTotoznostRequest" />
            <wsdl:output message="impl:overTotoznostResponse"
             name="overTotoznostResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="ServiceSoapBinding" type="impl:Service">
        <wsdlsoap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="overTotoznost">
            <wsdlsoap:operation soapAction="" />
            <wsdl:input name="overTotoznostRequest">
                <wsdlsoap:body use="literal" />
            </wsdl:input>
            <wsdl:output name="overTotoznostResponse">
                <wsdlsoap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ServiceService">
        <wsdl:port binding="impl:ServiceSoapBinding" name="Service">
            <wsdlsoap:address
            location="http://localhost:8080/PrezentaceSilverlightService/services/Service" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Obrázek 28: Deskriptor Webové služby pro Silverlight aplikaci